INTEGRATING VERSIONS, HISTORY, AND LEVELS-OF-DETAIL WITHIN A 3D GEODATABASE

G. Gröger, T. H. Kolbe, J. Schmittwilken, V. Stroh, L. Plümer

Institute for Cartography and Geoinformation, University of Bonn, Meckenheimer Allee 172, 53115 Bonn, Germany
{groeger, kolbe, schmittwilken, stroh, pluemer}@ikg.uni-bonn.de

KEY WORDS: 3D city models, 3D GIS, geodatabase, LOD, multi-representation, planning versions, history, workspaces

ABSTRACT:

The sustainable management and continuation of virtual 3D city models puts specific demands on 3D geodatabases. These demands result from the need to manage three different dimensions of multiple representation: Levels-of-detail, concurrent planning versions, and history over time. In this paper we propose a concept for the integration of all three aspects in the framework of spatial relational database management systems. User applications need to see resp. work with different states of the database content, e.g.. the current state "as-built" or the current state plus certain approved plannings. It is shown how the workspace manager of the RDBMS can be extended by structures, which allow to combine these versions in so called City Model Aspects. Specific spatial integrity checks are introduced in order to avoid conflicts between spatially disconnected concurrent plannings.

1. INTRODUCTION

The sustainable management and continuation of virtual 3D city models puts specific demands on 3D geodatabases. First, the content of 3D city models evolves over time as buildings and other objects are added resp. changed or deleted reflecting urban development. However, planning and documentation tasks often need to have access to the state of the model at any point of time in history. Second, urban planning needs to be able to manage different versions and competing (re-)designs for the same geographic areas. Third, 3D city models typically have a multi-scale representation, i.e. all entities can be stored in different levels of detail (LOD) regarding both geometry and semantic granularity.

All three aspects are distinct, and from a logical perspective "orthogonal" facets of multi-representation (cf. Kolbe & Plümer 2004). In the following, we analyze the specific properties of each facet, before we show how the three facets can be modeled in terms of the concepts of spatially enabled relational database management systems (Geo-RDBMS) in an integrated way.

We exemplify the concept by the realization of a unified, multi-scale data model for 3D city models on top of an object-relational 3D geodatabase implemented in Oracle Spatial (cf. Gröger et al. 2004). The employed data model has been developed by the Special Interest Group 3D (SIG 3D) of the initiative Geodatainfrastructure North-Rhine Westphalia (GDI NRW) over the last 3 years (see Kolbe & Gröger 2003). It is also the starting point for the development of CityGML, the GML3-based standard for the interoperable access of 3D city models (cf. Kolbe et al. 2005).

2. FACETS OF MULTI-REPRESENTATION

3D city models are often classified with respect to their resolution and generalization level. In many existing models, discrete levels of detail (LOD) are distinguished. The SIG 3D model comprises five LOD, starting with the coarsest LOD 0 describing a regional model, and ending with the fully detailed LOD 4, which comprises both the exterior and interior modeling of city objects like buildings (Kolbe & Gröger 2003). Representations differ both with respect to geometry and to semantic structuring. The specific characteristic of LOD is that multiple representations of the same object are simultaneously valid.

The continuation process of city models implies the second facet of multi-representation. Spatial objects change over time, and therefore have different representations at different points in time (in any LOD!). This aspect is especially important with regard to 3D cadastres, where the complete history of the city objects has to be maintained. In contrast to the LOD case above, at any time exactly one representation of an object (in all LOD) is valid.

The third facet of multi-representation results from urban planning. A planning scenario typically comprises competing designs reflecting alternative concepts and ideas. Thus, some objects may exist in concurrent versions. However, these parallel alternatives may not be used or visualized simultaneously. The problem gets even more complicated by the fact that there exist different construction projects at the same time. Users and planners need to see or work with a specific view on the 3D city model, in which an arbitrary combination of a selected version for each different scenarios is reflected. Since these alternative representations may exist in different LOD and at any point in time, versions add another dimension to the problem of multiple representations.

3. VERSION AND HISTORY MANAGEMENT USING WORKSPACES

A well-known concept for the management of versioned data in DBMS is the *workspace*. A workspace provides a virtual copy of the state of a database (or another parent workspace), in which modifications are carried out locally and stay separated from other workspaces (cf. Katz 1990). The changes may be merged into the original database content resp. the parent workspace at a later point in time. Conflicts occur when the same object is changed in different workspaces, and most database management systems support the detection of such conflicts and the process of solving them by the user. Workspaces typically are used to support long-term transactions in databases, which may take many months up to several years.

The concepts of versions and workspaces are quite mature and have their origins in the 80s. They have been developed in the context of CAD databases in order to enable concurrent work on different parts of a VLSI integrated circuit design (Katz 1990). The Oracle workspace manager now incorporates most of these concepts into the Oracle RDBMS (Agarwal et al. 2003, Oracle 2003). Versioned tables are augmented by columns that denote the version and workspace of each data tuple together with the date and time of every update. Database views and triggers then are constructed for each version, which preserve the original table structure but show only the database content that belongs to the respective version. Batty (2002) describes this approach as *shallow version management*, which has the advantage of being almost transparent to applications. Another prominent version management concept is based on explicit checkouts and checkins of portions of the geodatabase for the geographical areas to be updated (Katz 1990). However, this approach is not feasible with respect to the need for integrated views of the whole city showing the impact of the different ongoing plannings simultaneously.

In the following we employ the workspace concept to support the planning processes for 3D city models. Since only the changes or new added or generated objects are recorded by the DBMS, redundant storage of planning regions is avoided. The application of this concept for the management of a single planning project is straightforward. Each project typically comprises different competing planning versions for the same region, accomplished by different planners or architects. Since a workspace holds exactly one state of the city model, a planning project would have to be represented by one workspace for each competing version. After the planning process is completed, one single version is chosen to be implemented and is submitted (posted) to the parent workspace resp. the root workspace.

3.1 Handling of Conflicts

Problems arise due to the fact, that city development generally covers a multitude of simultaneous, but independent planning projects. In each project, one planning version is selected finally and implemented after the process is completed. Thus the changed or newly generated objects have to be updated or inserted in the parent or root workspace. In this process of merging different versions, conflicts may occur, if the same object is modified in different alternatives or versions. These conflicts, of course, have to be avoided.

Different planning projects usually are related to different regions, which are spatially disjoint. If all updates and insertions are restricted to the region of the corresponding planning activity, no conflicts occur when the different alternatives will be merged. However, it has to be ensured by the database that in fact all changes are restricted to the planning region. To perform this task, we propose an approach which stores the planning region explicitly as a spatial attribute of a planning activity and which employs the database concept of *triggers* (Ullman 1988). A trigger is a procedure, which is executed automatically by the database management system, when a update or insertion on a field of a database table occurs and a certain condition is met by the database state. The kind of update or insertion and the condition is given explicitly in the trigger definition. For example, a trigger may specify that an update of the field 'height' of a table 'building' is not allowed, when the new value of this field is negative.

To apply this concept for conflict avoidance between planning activities, a straightforward approach would be to define triggers on all update operations (insertion, modification, and deletion) on geometry fields of spatial features. These would check whether the modified object is inside the planning region associated with the corresponding planning. If this is not the case, the modification would be rejected. This check can be done very efficiently using the spatial indices and query filters of the database.

The approach described above is sufficient if no spatial object touches more than one planning region. However, there exist a number of object types like highways and rivers for which it is very likely that they touch more than one planning region (see fig. 1). Although these objects might be modified only locally in the different planning activities, the changes cause conflicts in the database. This is due to the fact, that the same object that contains the geometry is modified more than once.

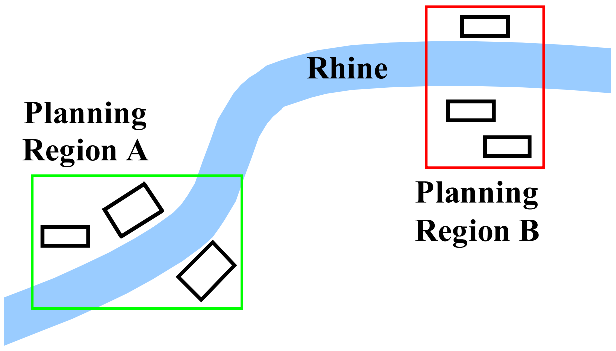


Fig. 1: One spatial object (the Rhine river) touching two disjoint planning regions (A and B)

The traditional approach to deal with this problem is to split large objects into parts, each part touching only a single planning region. However, object identity and object homogeneity would be affected, and all references to the objects would have to be updated accordingly.

In the following we propose an approach with a higher granularity, which maintains objects as a whole. The crucial point is the definition of the trigger condition. While in the first approach the condition checks whether the whole modified object is inside the planning region, the new condition considers only the actual geometric modification and checks whether it is inside the planning region. Fig. 2 illustrates this approach.

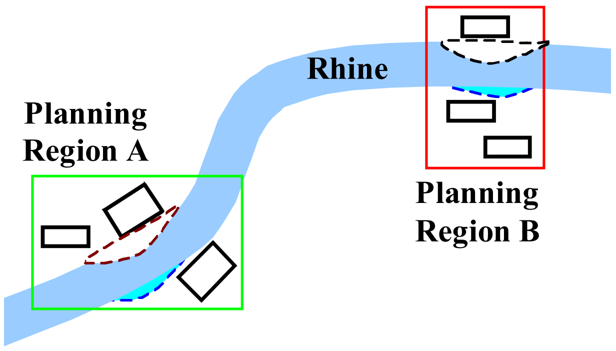


Fig. 2: Two modifications (hatched areas and dashed lines) of the same object (Rhine river), which are located in different, disjoint planning regions (A and B)

The object river Rhine is modified twice, in region A and in region B. The first approach would reject those changes, since the modified object touches region A and B. Since each geometric modification is strictly inside a planning region, our new approach admits this situation.

The process of merging versions of different planning regions is more difficult to handle and has to be adapted. Since the same object may have different geometric modifications in different regions, the two or more new geometries have to be merged. In principle, the functionality of spatial databases may be used for this task. Oracle Spatial, for example, offers various functions to intersect (∩) or merge (∪) two geometries or to subtract one geometry from another (–). Using these functions the merging process can be formalised as follows:

Rhine.geometry – A.extent – B.extent
∪
Rhine.geometry[Version A] ∩ A.extent
∪
Rhine.geometry[Version B] ∩ B.extent

First, the areas of the extents of all affected planning versions are subtracted from the original river geometry. The result represents the untouched portions of the river geometry. In the next step, the river geometry is clipped for every planning version to the extent of the according planning region. Finally, all resulting geometries are merged by the union function into one river polygon again. This single polygon combines the geometric modifications of all plannings involved.

Whereas the example illustrates the conflict management in a 2D scenario, the concept covers the 3D case as well. However, since support for 3D geometric primitives is currently still lacking in geodatabase management systems, no predefined database functions can be used to implement the merging procedure for 3D immediately (Zlatanova et al. 2002). First steps of the integration of 3D primitives have been presented by (Stoter & Oosterom 2002), but unfortunately the functionality is not available in commercial DBMS yet.

3.2 Aspects: Integrated Views over different Planning Regions

Users and city planners are interested in seeing and working on an integrated view of the 3D city model, in which a user-defined or predefined combination of selected designs for the distinct planning projects is reflected. We call any such specific view a *city model aspect*. City model aspects are used to define different 3D city model products. For example, one specific city model aspect may show the current state of the city model, a second the current model plus approved versions of ongoing planning projects, and a third the current model plus all designs from a specific architect for the different planning projects in which he is involved.

Since each combination has to be represented by one workspace and any combination may be possible, there is a potential combinatorial explosion of needed workspaces. The solution for the described problem lies in defining an appropriate structuring schema for workspaces and the application of so-called *multi-parent workspaces*. Multi-parent workspaces are workspaces which have more than one predecessor (Agarwal et al. 2003, Oracle 2003). They integrate the different states of the parent workspaces, provided that no conflicts occur on the tuple level.

Now, for each current planning region, a set of workspaces is derived from the 3D reference model. Each workspace contains a different planning version of the corresponding region and is stamped by the name of the planning region and the version identifier. For each city model aspect resp. individual user configuration, a multiparent workspace is generated on demand, integrating the desired versions for each region. The information about the existing planning projects and the contained competing versions is kept in two additional tables (see fig. 3).

We have implemented the concept using the RDBMS Oracle 10g and the Oracle Workspace Manager (OWM). The OWM allows any table to be put under the control of the versioning mechanism. Then, an arbitrary number of workspaces can be created, where each maintains its own state of the versioned tables. Since versioned tables virtually retain their original relational schema, the usage of workspaces is almost transparent to applications. An application only has to select the appropriate workspace before starting to work with the database.

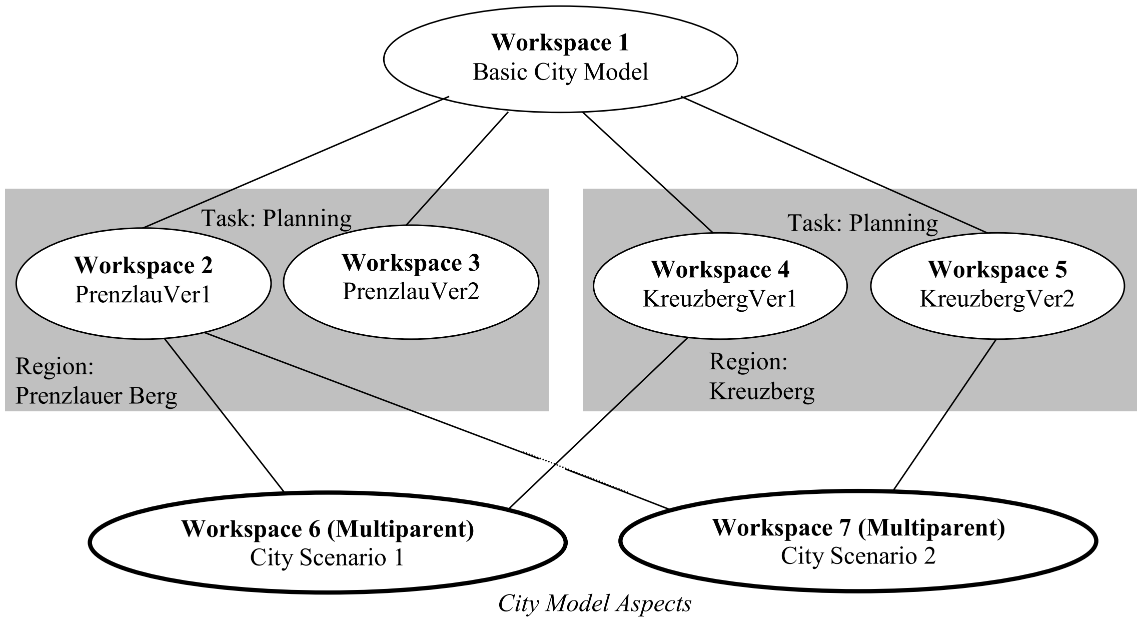
The workspace concept is also being used for the management of history. Since updates on versioned tables are stamped with time and date, the database access cannot only be focused on explicit versions but also on specific points of time in history. This includes history over the different, concurring planning versions – even those versions, which have not been selected for realization in the past.

4. INTEGRATION OF MULTIPLE LEVELS-OF-DETAIL

The representation of multiple levels-of-detail of the same spatial phenomenon differs from the other two facets of multi-representation, versions and history, on a conceptual level. This has strong impacts on the way to implement multiple LOD in a spatial database. Across different versions and points in time, the schema of a database remains constant, allowing applications to access the database in a transparent way. Across multiple LOD, however, this is not the case in general. For example, a coarse LOD may represent a building by a single class on a semantical level, and by a simple block geometrically. A more detailed representation provides several thematic classes for building parts like balconies, chimneys and roofs, and a more elaborated geometry with solid and surface geometries (Köninger & Bartel 1998, Coors & Flick 1998, Kolbe & Gröger 2003). Thus the database schemas in both LOD differ. As a consequence, accessing different LOD may not be transparent to applications, which must be aware of the specific schema in the desired LOD. Thus the problem of representing multiple LOD cannot be solved on the level of database mechanisms, but on a conceptual modeling level, i.e. the corresponding LOD must be reflected in the UML diagrams from which the database schema is derived.

Our approach to distinguish multiple LOD is to tag or stamp the UML classes and relations by the corresponding LOD identifier (cf. Vangenot et al. 2002). In our model (Kolbe et a. 2005), a thematic object may have different spatial representations in different LOD. These spatial properties are based on a geometric-topological model. It is implemented using a hybrid representation using the Oracle Spatial data type SDO_GEOMETRY on the one hand, and an explicit relational modeling of topology on the other hand. The former provides efficient spatial indexing mechanisms, while the latter is needed in order to maintain topological consistency. A detailed description of the model is given in (Gröger et al. 2004).

The number of classes in our model increases with growing degree of detail, by specializing classes into sub-classes or by adding more detailed thematic aspects, but no classes are



Workspace Structure Table

Workspace No	Workspace Name	Region	Extent (Polygon)	Starting Date	Termination Date
WS 2	PrenzlauVer1	Prenzlauer Berg	....	12-07-2002	12-12-2003
WS 3	PrenzlauVer2	Prenzlauer Berg	....	07-06-2002	12-11-2003
WS 4	KreuzbergVer1	Kreuzberg	....	05-03-2004	
WS 5	KreuzbergVer2	Kreuzberg	....	12-04-2004	

City Model Aspect Table

Workspace No	Workspace Name	Parent Workspaces
WS 6	City Scenario 1	WS 2, WS 4
WS 7	City Scenario 2	WS 2, WS 5

Fig. 3: Simple example for the structuring of different planning versions and their integration into user defined *city model aspects*. The topmost workspace contains the current state of the city model. There are planning activities in different regions, where the designs are represented in concurring workspaces. The integrated views (City Scenario 1 & 2) are provided by multiparent workspaces which incorporate selected planning versions. Please note, that each workspace does represent the whole 3D city model with only regional modifications. The tables keep track of the different versions and are needed for the construction of the *city model aspects*, i.e. the multiparent workspaces.

dropped. When classes are aggregated to or assimilated by different, coarser classes, and these generalized classes are omitted in more detailed LOD, the issue of representing LOD becomes more complex. The relations between different classes in different LOD representing the same spatial phenomenon must be recorded explicitly (c.f. Vangenot et al. 2002 or Kolbe & Gröger 2003). Relations with specific semantics must be introduced to guarantee that queries or visualizations are consistent, i.e. that no spatial phenomenon is considered more than once. Corresponding rules operating on theses specific relations may be found in (Kolbe & Gröger 2003).

5. CONCLUSIONS AND OUTLOOK

Th presented concepts show how the three different aspects of multi-representation can be mapped to spatially enabled RDBMS. Whereas history and version management is handled almost transparently to the user, the representation of levels-of-detail affects the data model and thus has to be treated by the application on top of the geodatabase. City model aspects are used to define application-relevant combinations of versions. By maintaining the extent of planning regions, conflicts between concurrent versions can be assessed and resolved not only on a database tuple level, but more specific on a geometry parts level.

Our prototypical implementation of the 3D geodatabase system is based on Oracle Spatial 10g and demonstrates the feasibility of the concept. In fig. 4 a small city scene is shown, where two distinct planning regions are depicted (one on the left and one on the right side). For each planning, two workspaces are created to hold two concurrent designs respectively. Fig. 5 shows two competing designs for the left hand planning, while fig. 6 is focussed on the right hand planning. In order to view a

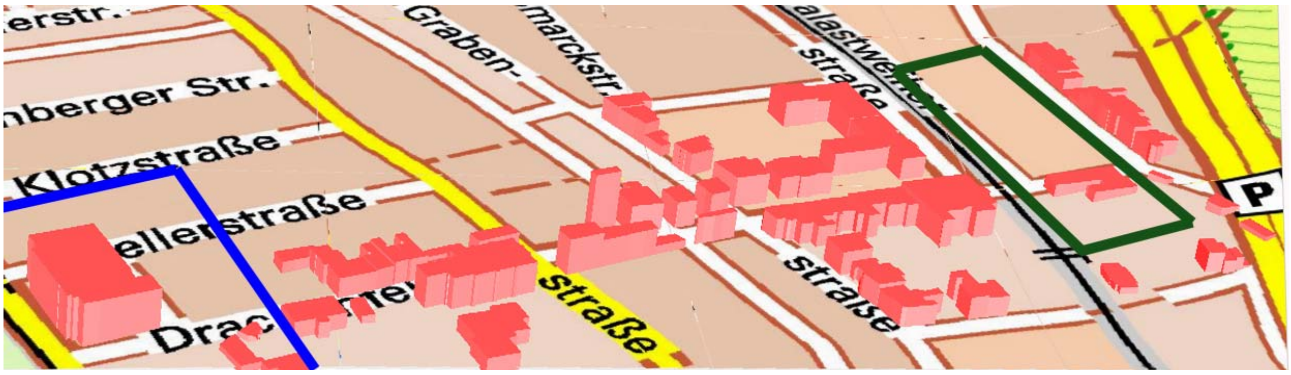


Fig. 4: Portion of a 3D city model. Two planning regions are described by their spatial extent: one on the left (blue) and another on the right (dark green). Each planning may comprise several competing alternatives as versions. Each alternative is assigned to one workspace which stands for a virtual copy of the 3D city model.

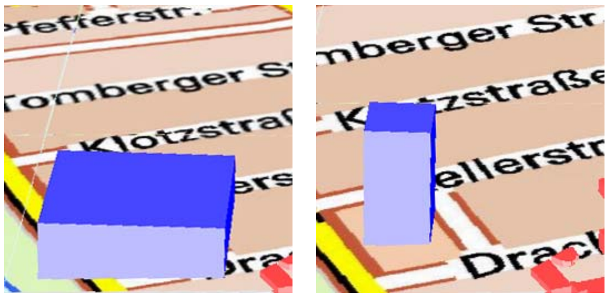


Fig. 5: Two versions of the "blue" planning. Each version is stored in a separate workspace.

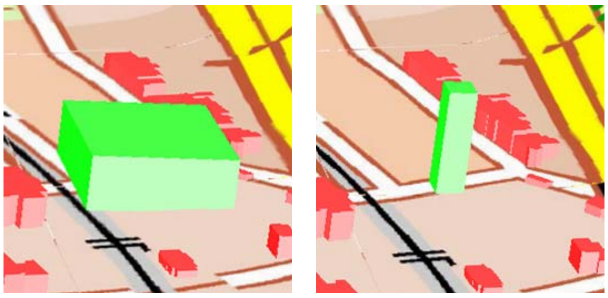


Fig. 6: Two alternatives of the "green" planning. Please note, that each workspace contains the whole dataset.

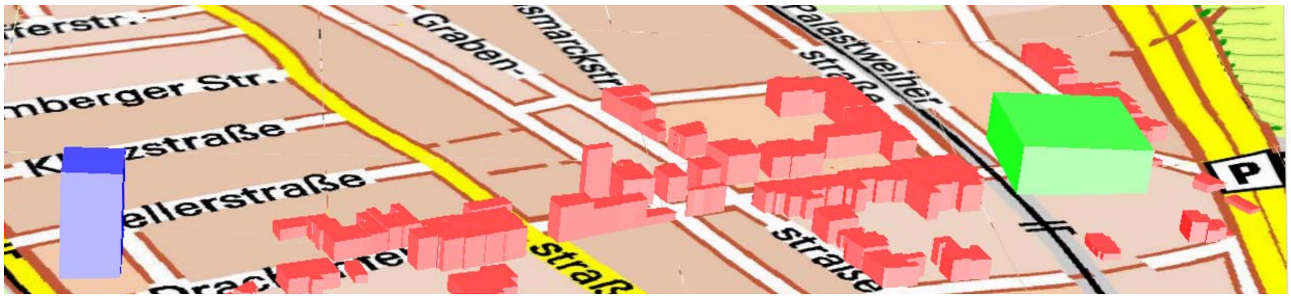


Fig. 7: A city model aspect was defined to show the combination of alternatives of different plannings. Here the tall version of the "blue" planning and the broad version of the "green" planning have been combined. A city model aspect is a multiparent workspace whose parents are the workspaces associated to the planning alternatives (cf. fig. 3).

combination of specific designs, a city model aspect consisting of the 2nd alternative of the left hand and the 1st of the right hand planning is defined.

In the future, we intend to investigate how the conflict management rules can be incorporated into the workspace manager of the database. Thus triggers will be generated automatically on all tables with spatial attributes when new workspaces are created.

ACKNOWLEDGEMENTS

Parts of this work have been done within the EU-funded project "Geodata management for the administration of Berlin – 3D VR model for investors and enterprises" in cooperation with the state administration and business development Berlin.

REFERENCES

Agarwal S, Arun G, Chatterjee B, Speckhard B, Vasudevan (2003): *Long transactions in an RDBMS*. In: Proceedings of the 26th GITA Conference 2003 in San Antonio, Texas

Batty PM (2002): *Version Management Revisited*. In: Proceedings of the 25th GITA Conference, March 17-20, 2002 in Tampa, Florida

Coors V, Flick S (1998): *Integrating Levels of Detail in a Web-based 3D-GIS*. Proc. 6th ACM Symp. on Geographic Information Systems (ACM GIS 98), Washington D.C., USA.

Gröger G, Reuter M, Plümer L (2004): *Representation of a 3-D City Model in Spatial Object-relational Databases*. In: Intern. Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. 34. Part B4 (Proc. of the XXth ISPRS Congress, Istanbul, Turkey).

Katz RH (1990): *Toward a Unified Framework for Version Modeling in Engineering Databases.* ACM Computing Surveys, Vol. 22, No. 4

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

Kolbe TH, Gröger G (2003): *Towards unified 3D city models*. In: Schiewe, J., Hahn, M, Madden, M, Sester, M (eds): Challenges in Geospatial Analysis, Integration and Visualization II. Proc. of Joint ISPRS Workshop, Stuttgart

Kolbe TH, Gröger G, Plümer L (2005): *CityGML – Interoperable Access to 3D City Models*. In: van Oosterom, P, Zlatanova, S, Fendel, EM, (Eds.) Geo-information for Disaster Management (Proc. of the first International Symposium on Geo-Information for Disaster Management GI4DM), Delft, The Netherlands, March 21-23, Springer.

Kolbe TH, Plümer L (2004): *Bringing GIS and CA(A)D Together - Integrating 3d city models emerging from two different disciplines.* GIM International, Vol. 18, No. 7, July 2004.

Oracle (2003): *Database Application Developer's Guide - Workspace Manager* (2003), 10g Release

Stoter JE, van Oosterom PJM (2002): *Incorporating 3D geo-objects into a 2D geo-DBMS.* In: Proc. FIG, ACSM / ASPRS, Washington D.C., April 2002

Ullman JD (1988): *Principles of Database and Knowledge-Base Systems*. Vol. 1, Computer Science Press.

Vangenot C, Parent C, Spaccapietra S (2002): *Modelling and Manipulating Multiple Representations of Spatial Data*. In: Proc. of the Symposium on Geospatial Theory, Processing and Applications, Ottawa, 2002.

Zlatanova S, Rahman AA, Pilouk M (2002): *3D GIS: Current Status and Perspectives.* In: Proc. of the Symposium on Geospatial Theory, Processing and Applications, Ottawa, 2002.