An importer of virtual 3D City Models datasets into a spatiotemporal database

Wagner Franchin[1], Alexandre Carvalho[1], José Moreira[2], A. Augusto de Sousa[1] and Cristina Ribeiro[1]

[1] INESC Porto, Campus da FEUP, Rua Dr. Roberto Frias, 378, 4200-465 Porto, Portugal - [wjf, alexandre.carvalho, aasousa, mribeiro]@inescporto.pt
[2] IEETA, Campus Universitário de Santiago, Universidade de Aveiro, 3810-193 Aveiro, Portugal - jose.moreira@ua.pt

Abstract. Geographical Information Systems (GIS) have an important role in a wide range of application domains, both for the management of spatial data and as a decision support tool. For this reason virtual 3D city models are becoming increasingly complex with respect to their spatial, thematic structures and temporal aspects. In this paper we present Importer CityGML, a tool for parsing and importing XML datasets into a spatiotemporal database prototype system.

Keywords: Spatiotemporal Databases, CityGML, Virtual 3D City Models

1 Introduction

The need of 3D City Models is growing, especially for organizations involved in urban and landscape planning, cadastre, real estate, utility management, geology, tourism, army, etc. Geographical Information Systems (GIS) have an important role in a wide range of application domains, both for the management of spatial data and as a decision support tool. However, current GIS have been developed to deal efficiently with the current state of spatial data, but they do not include temporal features to deal with previous states about spatial information.

To cope with this limitation, Carvalho et al. [1] present a spatiotemporal database prototype system (*Action!*), developed on the top of an Oracle database management system and TimeDB [2], for efficient representation, querying and visualization of time-evolving urban information.

Further, CityGML [3] is a general information model for representing geovirtual 3D environments such as virtual 3D city models. It introduces classes and relations for topographic objects of urban environments and regional models.

This paper presents a tool to transfer information from a CityGML files into *Action!*, in order to benefit from the spatiotemporal querying and visualization capabilities of this system.

2 Related work

Generating virtual city environments has already been addressed in several works. The initial methods were based on computer aided architectural design where detailed measurement of the geometry was regarded as essential. Kolbe et al. [3] proposing a new model based on GML to describe data for any kind of city, Hagedorn and Dollner [4] describing an approach to visualize and analyze CAD-based 3D building information models (BIM) within 3D virtual city models, and Google SketchUp [5] allowing creating, visualizing and modifying 3D representations, are notable examples.

CityGML [3] is an open data model structure and standardized code based on XML for storing and exchanging virtual 3D city models. The common information model behind CityGML defines classes and relationships for the most relevant topographic objects in cities and regional models with respect to their geometrical, topological, semantic and appearance properties. The thematic model of CityGML considers thematic fields like Digital Terrain Models, sites (i.e. buildings), vegetation (solitary objects and also areal and volumetric biotopes), water bodies, transportation facilities or city furniture.

Finally [6] Kolbe et al. present a 3D geo database for CityGML. The CityGML data model is mapped into a relational database schema and an import/export tool was realized for processing of CityGML and GML structures. The system is not prepared to manage temporal information.

3 Importer CityGML

Importer CityGML is a tool for parsing and importing CityGML datasets into *Action!*. The system was developed in Java and uses classes and interfaces provided by XML Beans.

The **data model** defined in *Action!* is based on the CityGML schema and classes. As CityGML data files may hold objects with a null identifier, the primary key of each relation in *Action!* is a unique identifier generated automatically.

The CityGML class *Building* is mapped into a relation also named *Building*, to store the details about constructions, such as buildings and houses. The *OuterBuildingInstallation* relation stores information about the outer parts of buildings, for example, cash, chimneys or staircases. *BoundarySurface* relation holds information regarding the parts of a building, e.g. inner and outer walls, ceiling or roof.

The CityGML classes *CityFurniture*, *Generics*, *LandUse*, *Transportation*, *Vegetation* and *WaterBody* were aggregated into a single relation *ThematicObject*, as they share the same set of attributes. The *CityObjectGroup* class is represented in the database by the *CityObjectGroup* relation. This relation stores information about groups of objects in a city, for example, an university. *GroupMember* relation identifies the elements composing a group of objects, which can be groups of buildings or thematic objects.

The relation *Geometry Object* stores the geometric information and the level of detail of all city objects in CityGML data files. It also includes a temporal component defined by two additional attributes (start date and end date), denoting a valid time interval, and an attribute of type BLOB (Binary Large Object) to store all elements that make up a geometry in a JME (Java Monkey Engine [7]) compatible format, used by the *Action!* visualization tool.

SurfaceMember relation stores information about the type of geometry surfaces (Polygon, TexturedSurface, OrientableSurface or SurfaceMember) and *CoordinateGeometry* is the relation holding geometric coordinates of each surface. The coordinates are stored in an Oracle Spatial column of type Sdo_Geometry and so, they ready to be used by Oracle Spatial's operations.

To make the import into the database easier, each relation in the database model has a corresponding Java class. During the **parsing**, the information contained in the CityGML file is stored in a list of objects in Java. For instance, each building has a list of objects composed by geometry objects, boundary surfaces or outer building installations. Each boundary surface has another list of objects composed by geometry objects and so on.

After the parsing phase, the system has all file information in memory (in a city object list) and starts **importing data** into the database. The first relation to be filled is *CityModel*. After that, *Building*, *ThematicObject*, *CityObjectGroup* and *Appearance* are ready to be filled (see Fig. 1).

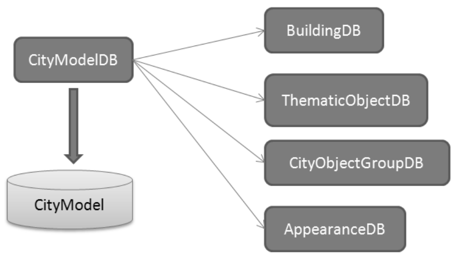

Fig. 1. CityModelDB class transfers data to CityModel table and passes the control to another class.

For example, when there is a Building in city object list, the system first imports the corresponding information into the *Building* relation and goes to the next element in this building object list. The next element can be a *GeometryObject*, *BoundarySurface* or *OuterBuildingInstallation*. If the system finds at this moment a *BoundarySurface* or an *OuterBuildingInstallation*, for example, the information is sent to the corresponding relations and the processing continues with the next object in the list.

Figure 2 illustrates a virtual 3D city at different dates, imported from CityGML into *Action!*. The original CityGML file was edited before importing to insert the temporal information.

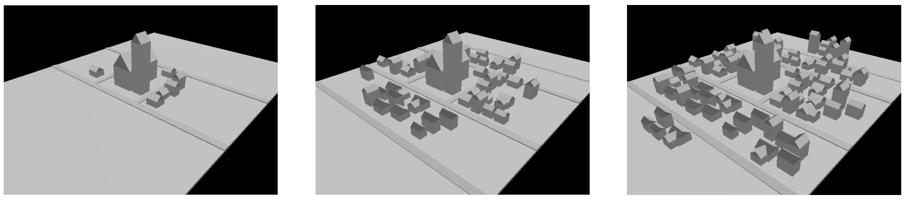
(a) 1300 (b) 1700 (c) 2000

Fig. 2. Figures **(a)**, **(b)**, **(c)** shows the virtual in the year of 1300, 1700 and 2000, respectively.

4 Conclusion and future work

This paper presents the Importer CityGML, a tool for parsing and importing CityGML datasets into a spatiotemporal database system. The system has initially been realized as an Oracle 10G R2 Spatial relational database schema. After parsing the XML file, each CityGML object and its corresponding information are imported to the tables into the relational database. The main implementation goals are to achieve efficient storage, fast processing of CityGML and spatiotemporal management capability.

Future work will extend the system to parse and import other CityGML classes which have not been implemented, for example, ParameterizedTexture.

References

1. A. Carvalho, C. Ribeiro, and A. Sousa, "A Spatio-Temporal Database System Based on TimeDB and Oracle Spatial" in IFIP International Federation for Information Processing. Springer, 2006, pp. 205: 11-20.
2. Steiner, A., "A Generalization Approach to temporal Data Models and Their Implementations", Phd, Zurich, 1998.
3. T. Kolbe, G. Groger, and L. Plumer, "Interoperable Access to 3D City Models". in First International Symposium on Geo-Information for Disaster Management. Delft, Netherlands: Springer, March 21-23 2005.
4. B. Hagedorn and J. Dollner, "High-level web service for 3d building information visualization and analysis", in GIS 07: Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems. New York, NY, USA: ACM, 2007, pp. 1-8.
5. "Google Sketchup." [Online]. Available: http://sketchup.google.com/ (July 15th, 2009)
6. T. H. Kolbe, G. Knig, C. Nagel, and A. Stadler, "3d-geo-database for citygml - version 2.0.1", Institute for Geodesy and Geoinformation Science Technische Universitt Berlin, Tech. Rep., 2009.
7. "Java Monkey Engine." [Online]. Available: www.jmonkeyengine.com/ (July 15th, 2009)