ABSTRACT
Nowadays, many state-of-the-art user interfaces to complexly structured data collections require appropriate information visualization methods. With the increase in graphics performance available on client systems, methods for three-dimensional visualization can be applied more easily. The conceptual approach presented in this paper dynamically generates three-dimensional VRML scenes from information stored in database systems. To achieve this, VRML is extended with new nodes for supporting the server side generation of VRML scenes and for supporting trigger mechanisms. An implementation of the introduced concept is introduced by means of an object relational database system which is extended with a new VRML data type and its related functions. Altogether, these concepts and software components provide a platform for server-based information visualization mechanisms.

Keywords
Information Visualization, VRML, Object relational database system, Trigger

1. Introduction and Motivation
The growing number of complexly structured data collections accessible over the global information network increasingly requires the support of appropriate visualization methods. Visualization is needed to support information retrieval by browsing and searching [4][8], as well as data mining tasks for naive end users. The available graphics performance of network client systems is steadily growing; methods for three-dimensional visualization can be applied more and more easily.

Three-dimensional visualization systems are applicable to various application domains. One example is the visualization of large data collections within information systems [10]. Another important application area is electronic commerce. Three-dimensional visualizations are a powerful means to increase the cognitive effectiveness of virtual shopping malls, virtual libraries, or virtual galleries for naive end users. Information systems to support large events, like trade fairs or congresses, can benefit from three-dimensional visualizations [9]. A virtual-reality overview of a trade fair can, e.g., display information about all companies exhibiting their products on stalls. Visitors can plan their routes through the trade fair by walking through a three-dimensional visualization of the trade fair hall, long before the actual visit takes place. Exhibitors can get an impression of their stand and its position among competitors’ stalls. Furthermore, the visualization of the hall and stands can be used as a metaphor to access further information about the exhibitors and their products, or to send requests for meeting appointments to the exhibitors’ staff members.

2. The Current Situation
The Virtual Reality Modeling Language (VRML) is widely used on the Internet to describe three-dimensional scenes [13]. However, today, VRML is mostly used to handle static scenes. After loading a scene, the user starts his exploration of the chosen virtual world always in the same state [17]. Changes applied to a scene are lost after a session. The virtual worlds are limited in size and complexity, since the complete scene information has to be held within the VRML browser. Although a dynamic generation of scenes could support different levels of levels of detail [18][3], even in advanced applications of VRML, the scenes have always the same complexity regardless of the capabilities of the browser. The dynamic generation of scenes depending on different access rights of different users is not possible. Tools to support the dynamic generation of VRML scenes are rare since their development turns out to be a costly effort [1].

3. An Approach Towards Supporting Dynamic Scene Generation
Today, the dynamic generation of HTML pages is a standard functionality of all commercial database systems. This feature
has been proven to be a very effective and practical approach to support the (two-dimensional) visualization of information stored within database systems. In this paper, we outline how a similar functionality can be realized to dynamically generate VRML scenes from a database management system (DBMS). This approach overcomes many of the limitations of static VRML scenes, by exploiting the persistence, scalability and security mechanisms of database management systems [2]. In addition, it also provides a direct way to efficiently generate three-dimensional visualizations from existing information collections.

In order to provide advanced storage and management support for VRML scenes, the permanent and structured storage of VRML scenes or scene components is a basic requirement, together with related metadata. As a result, the DBMS has to be able to treat the VRML data type like any other built-in data type, and support search and manipulation operations on it. Such a VRML data type allows, for example, to store all stands of a trade fair as VRML scenes together with the company name and the position in the hall. It is possible to select a subset of all stands with certain properties and merge them in a new scene, e.g., in order to display all stands that have already been rented to customers.

Since most of the information stored in a database will not be three-dimensional scene data, a DBMS supporting three-dimensional visualization must be able to generate new VRML scenes, both from existing operational business data represented by conventional data types and from existing multimedia data represented by specialized media data types. Within the trade fair system, statistical data about expected visitor flows in a hall could be visualized by an arrow diagram, where the arrow size is proportional to the expected visitor flow.

The VRML client has to be able to directly read and write the DBMS from within a VRML scene. For more advanced interaction modes in multi-user environments, this mechanism needs to be complemented by an event handling system. This allows to signal a change in a scene to all other users actively working on the same scene. The VRML event handling of the other users can then react by appropriately updating the scene, e.g. by reloading a part or the whole scene.

The most important benefit of the above-outlined approach is that by means of storing VRML scenes within a DBMS, we achieve persistence of changes to scenes. Furthermore, the multi-user access control enables the sharing of VRML data among multiple users, thus we move from isolated, static scenes to shared spaces of dynamically generated three-dimensionally visualized information. Scalability is achieved by loading and generating scenes and scene components dynamically either at loading time or at run time. The corresponding loading/generation schemes can be determined both by physical characteristics of the VRML scene and the logical structure of the application. Controlled access to scene data is supported by the security and view mechanisms of the underlying DBMS. For example, in the "trade fair" scenario, one might display the rent of a stand only to authorized staff members or customers but not to visitors.

4. VRML Extensions

The implementation of the approach introduced above requires extensions to the database management system, the VRML language and the underlying processing of the VRML code [14] [15].

First of all, VRML is extended with a database-oriented node which is used as a so-called 'server side include' (SSI) mechanism. This means, that the VRML scene is parsed on the server side within the DBMS in order to expand the SSI node and replace it with other, standard VRML nodes. Therefore, the SSI nodes themselves will never be sent to the client.

In order to provide advanced storage and management support for VRML scenes together with the company name and the position in the hall, it is possible to select a subset of all stands with certain properties and merge them in a new scene, e.g., in order to display all stands that have already been rented to customers.

Since most of the information stored in a database will not be three-dimensional scene data, a DBMS supporting three-dimensional visualization must be able to generate new VRML scenes, both from existing operational business data represented by conventional data types and from existing multimedia data represented by specialized media data types. Within the trade fair system, statistical data about expected visitor flows in a hall could be visualized by an arrow diagram, where the arrow size is proportional to the expected visitor flow.

The VRML client has to be able to directly read and write the DBMS from within a VRML scene. For more advanced interaction modes in multi-user environments, this mechanism needs to be complemented by an event handling system. This allows to signal a change in a scene to all other users actively working on the same scene. The VRML event handling of the other users can then react by appropriately updating the scene, e.g. by reloading a part or the whole scene.

The most important benefit of the above-outlined approach is that by means of storing VRML scenes within a DBMS, we achieve persistence of changes to scenes. Furthermore, the multi-user access control enables the sharing of VRML data among multiple users, thus we move from isolated, static scenes to shared spaces of dynamically generated three-dimensionally visualized information. Scalability is achieved by loading and generating scenes and scene components dynamically either at loading time or at run time. The corresponding loading/generation schemes can be determined both by physical characteristics of the VRML scene and the logical structure of the application. Controlled access to scene data is supported by the security and view mechanisms of the underlying DBMS. For example, in the "trade fair" scenario, one might display the rent of a stand only to authorized staff members or customers but not to visitors.

The implementation of the approach introduced above requires extensions to the database management system, the VRML language and the underlying processing of the VRML code [14] [15].

First of all, VRML is extended with a database-oriented node which is used as a so-called 'server side include' (SSI) mechanism. This means, that the VRML scene is parsed on the server side within the DBMS in order to expand the SSI node and replace it with other, standard VRML nodes. Therefore, the SSI nodes themselves will never be sent to the client.

In order to provide advanced storage and management support for VRML scenes, the permanent and structured storage of VRML scenes or scene components is a basic requirement, together with related metadata. As a result, the DBMS has to be able to treat the VRML data type like any other built-in data type, and support search and manipulation operations on it. Such a VRML data type allows, for example, to store all stands of a trade fair as VRML scenes together with the company name and the position in the hall. It is possible to select a subset of all stands with certain properties and merge them in a new scene, e.g., in order to display all stands that have already been rented to customers.

Since most of the information stored in a database will not be three-dimensional scene data, a DBMS supporting three-dimensional visualization must be able to generate new VRML scenes, both from existing operational business data represented by conventional data types and from existing multimedia data represented by specialized media data types. Within the trade fair system, statistical data about expected visitor flows in a hall could be visualized by an arrow diagram, where the arrow size is proportional to the expected visitor flow.

The VRML client has to be able to directly read and write the DBMS from within a VRML scene. For more advanced interaction modes in multi-user environments, this mechanism needs to be complemented by an event handling system. This allows to signal a change in a scene to all other users actively working on the same scene. The VRML event handling of the other users can then react by appropriately updating the scene, e.g. by reloading a part or the whole scene.

The most important benefit of the above-outlined approach is that by means of storing VRML scenes within a DBMS, we achieve persistence of changes to scenes. Furthermore, the multi-user access control enables the sharing of VRML data among multiple users, thus we move from isolated, static scenes to shared spaces of dynamically generated three-dimensionally visualized information. Scalability is achieved by loading and generating scenes and scene components dynamically either at loading time or at run time. The corresponding loading/generation schemes can be determined both by physical characteristics of the VRML scene and the logical structure of the application. Controlled access to scene data is supported by the security and view mechanisms of the underlying DBMS. For example, in the "trade fair" scenario, one might display the rent of a stand only to authorized staff members or customers but not to visitors.
5. Database Extensions

Conventional relational database systems do not provide the functionality to directly support VRML scene data. Object-relational and object-oriented database management systems have been developed to support the extension by new data types, as they are required for new application domains, like multimedia or document handling. They provide the necessary mechanisms to support the modelling of VRML data types. Since most business data is stored in relational DBMS, we have chosen an object-relational DBMS, which can be understood as the next generation relational DBMS [11], for our development platform.

The core component to support VRML is the implementation of a VRML data type and its related functions. An object of type VRML data stores complete and valid VRML scenes. This is achieved by implementing the basic support for a simple VRML scene node. The functions defined for the VRML data type support type conversion, merging, import and export of predefined scenes as well as transformations of the complete scene. The function for scene merging can, to give an example, be used to process selections. If the result of a selection is a set of VRML scenes, they can be converted into a single scene before being sent to the client. If SSI nodes occur in the merged scene they must be expanded before the scene is sent to the client.

Finally, a trigger mechanism is implemented, such that trigger events are routed over an event server to the clients. The event server manages the active VRML clients and distributes the trigger events. By this architecture the database performance does not suffer when communication problems occur.

VRMLCreateWorld – Conversion of objects from SQL type VRML to a VRML scene
VRMLExpand – Conversion of objects from SQL type VRML to a VRML scene with expansion of templates.
FileToVRML, VRMLToFile – Import and export VRML scenes
VRMLSetTranslation (absolute), VRMLTranslate (relative) – Translation of a VRML scene
VRMLSetRotation (absolute), VRMLRotate (relative) – Rotation of a VRML scene
VRMLSetScale (absolute), VRMLScale (relative) – Scaling of a VRML scene
VRMLAddViewpoint – Adds a new viewpoint to the scene.

Figure 3 VRML data type functions

6. Conclusions and Future Work

With our approach, we achieve support for most of the requirements for dynamic information visualization support we have identified. Security and persistence are provided as a basic property of the DBMS. With the VRML data type and its corresponding functions the storage, retrieval and manipulation of VRML data is supported and implements server-side persistence of VRML data. Furthermore, the SSI mechanism provides support for the dynamic generation of VRML scenes from operational data like traditional relational data, multimedia data or VRML data itself. Multi-user environments and interaction-oriented applications can use the trigger mechanism for the communication between the database and the clients. Together, these components provide a platform for server-based information visualization. To close the interaction loop by enabling clients to actively interact with the server, some further development is currently under way. Currently, the described DBMS extension components can make persistent changes to VRML scene values, but for full persistency support updates of the client to the scene itself must be sent to the DBMS to initiate an update in the database. To this end, a so-called VRML SQL runtime node is currently under development. It will also be able to execute all sorts of SQL statements and to generate new VRML nodes from the corresponding results. The result processing mechanism that will be applied is the same as already implemented for the server-side include node. Together with the trigger mechanism, these developments will provide a sound base for the creation of multi-user VRML environments in the initially discussed application areas. Finally, server-side supported scalability is a problem area which is currently tackled. It will enable the generation of large VRML worlds regardless of the client’s hardware platform.

7. Literature


