Demonstration Proceedings 17th Int. Conf. on Data Engineering (ICDE), Heidelberg, Germany, 2001, pp. 15-16

DIVE: Database Integration for Virtual Engineering

Hans-Peter Kriegel[1], **Andreas Müller**[2], **Marco Pötke**[1] **and Thomas Seidl**[1]

[1]University of Munich, Institute for Computer Science
http://www.dbs.informatik.uni-muenchen.de
{kriegel, poetke, seidl}@dbs.informatik.uni-muenchen.de

[2]Volkswagen AG
http://www.volkswagen.de
andreas5.mueller@volkswagen.de

Abstract

This demonstration presents a spatial database integration for novel CAD applications as digital mockup or haptic rendering. The spatial data management and query processor is fully embedded into the Oracle8i server and has been evaluated in an industrial environment. Spatial queries on large databases are performed at interactive response times.

1 Introduction

In mechanical engineering, three-dimensional Computer Aided Design (CAD) is employed throughout the entire development process. From the early design phase to the serial production of vehicles or airplanes, thousands to millions of CAD files and associated documents are generated. Recently, a new class of CAD tools has emerged to support virtual engineering on this data, i.e. the evaluation of product characteristics without building a physical prototype. Typical applications include the digital mockup (DMU) [BKP 98] or haptic rendering of product configurations [MPT 99].

Engineering Data Management systems (EDM) organize the huge underlying CAD databases by means of hierarchical product structures. Thus, structural queries as "retrieve all documents that refer to the current version of the braking system" are efficiently supported. Virtual engineering, however, requires access to the product data by geometric predicates, such as "find all parts in the immediate spatial neighborhood of the disk brake" (cf. Figure 1). Unfortunately, spatial queries are not supported efficiently by common EDM systems. We present the DIVE architecture, a proposal to embed virtual engineering into the existing EDM infrastructure of the enterprise. In this work, we focused on the integration of interactive spatial data management into off-the-shelf object-relational database systems.

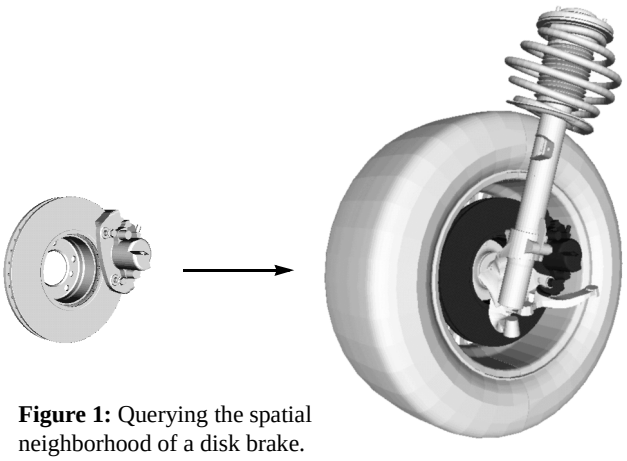

Figure 1: Querying the spatial neighborhood of a disk brake.

2 Spatial Data Management

The geometry of a part occupies a specific region in the product space. By using this region as a spatial key, related documents such as native CAD files, VRML scenes, production plans or meeting minutes may be spatially referenced. The key challenges in developing a robust and dynamic database layer for virtual engineering have been (1) to store spatial CAD data in a conventional relational database system and (2) to enable the efficient processing of the required geometric query predicates.

2.1 Object-Relational Storage

Only few spatial access methods have been designed to operate on the object-relational data model without any intrusive modifications or additions to the physical kernel of the database server. The Relational Interval Tree (*RI-tree*) [KPS 00] is a light-weight access method that efficiently manages extended data on top of any relational database system while fully supporting the built-in transaction semantics and recovery services. In [KPS 01] we have shown that its spatial application outmatches competing techniques with respect to usability and performance. We therefore use the RI-tree as spatial engine for the DIVE system.

The original parts are converted from the various native CAD formats to triangulated facets in VRML. Our approach utilizes 3D scan conversion on a regular grid to voxelize these geometries. To enable the generated voxel set to be used as a spatial key, it is transformed to an interval sequence on a space-filling curve and stored in the RI-tree. The redundancy and accuracy of each interval sequence can be controlled individually by size-bound or error-bound approximation.

2.2 Query Processing

The DIVE server maps geometric query predicates to region queries on the indexed data space. Our multi-step query processor performs a highly efficient and selective filter step based on the stored interval sequences. The non-spatial remainder of the query, e.g. structural exclusions, is processed by the EDM system. The current DIVE release contains filters for the following spatial queries:

- *Volume query*: Determine all spatial objects intersecting a given rectilinear box volume.
- *Collision query*: Find all spatial objects that intersect an arbitrary query region, e.g. a volume or a surface of a query part.
- *Clearance query*: Given an arbitrary query region, find all spatial objects within a specified Euclidean distance.

To demonstrate the full potential of our approach, we have integrated an optional refinement step for the digital mockup to compute intersections on high-accurate triangulated surfaces. A part resulting from the previous spatial query may be used as query object for the next geometric search. Thereby, the user is enabled to spatially browse a huge persistent database at interactive response times. The DIVE server also supports the ranking of query results according to the intersection volume of the query region and the found spatial keys. For digital mockup, this ranking can be refined by computing the shape and length of the intersection segments on the part surfaces. Thus, the attention of the user is immediately guided to the most relevant problems in the current product design.

3 System Architecture

Figure 2 presents the three-tier client/server architecture of the DIVE system. The client application runs on a conventional web browser and enables the user to specify spatial and non-spatial query conditions (1). The query evaluation is distributed to the DIVE and EDM servers (2). The DIVE server can be implemented on top of any relational database system, whereas extensible object-relational database systems facilitate the seamless embedding of complex spatial datatypes and operators. We have integrated the DIVE server

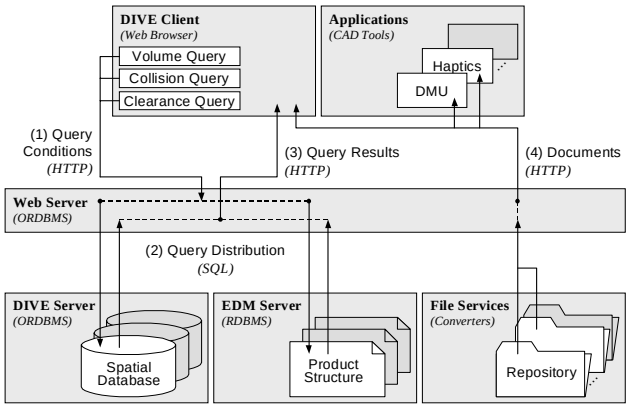

Figure 2: Processing a query on the DIVE system.

into Oracle8i by using PL/SQL and Java Stored Procedures. Therefore, the queries are simply submitted in the standard SQL syntax via Oracle's Net8 protocol. After completion of the spatial and structural filter steps and the optional query refinement, the query result is returned to the client as a table of document URLs (3). Finally, the browser may be used to display the contents of the corresponding documents or, alternatively, their content may be downloaded to a specialized application (4).

4 Industrial Evaluation

We have evaluated the DIVE server in an industrial environment on real product data. An installation on an Athlon/750 machine with IDE hard drives and a buffer pool of 800 KB performed average volume and collision queries in 0.7 seconds response time on a database containing 11.200 spatial keys (2 GB of compressed VRML data). Due to the logarithmic scaleup of our query processor, interactive response times can still be achieved for much larger databases.

References

[BKP 98] Berchtold S., Kriegel H.-P., Pötke M.: *Database Support for Concurrent Digital Mock-Up*. Proc. IFIP Int. Conf. PROLAMAT, Globalization of Manufacturing in the Digital Communications Era of the 21st Century, Kluwer Academic Publishers, 499-509, 1998.

[KPS 00] Kriegel H.-P., Pötke M., Seidl T.: *Managing Intervals Efficiently in Object-Relational Databases*. Proc. 26th Int. Conf. on Very Large Databases (VLDB), 2000.

[KPS 01] Kriegel H.-P., Pötke M., Seidl T.: *Interval Sequences: An Object-Relational Approach to Manage Spatial Data*. Proc. 7th Int. Symposium on Spatial and Temporal Databases (SSTD), LNCS, 2001.

[MPT 99] McNeely W. A., Puterbaugh K. D., Troy J. J.: *Six Degree-of-Freedom Haptic Rendering Using Voxel Sampling*. Proc. ACM SIGGRAPH Int. Conf. on Computer Graphics and Interactive Techniques, 401-408, 1999.