Database Research at IPSI

Editors: Erich Neuhold and Volker Turau
GMD, Integrated Publication and Information Systems Institute (IPSI)
Dolivostr.15, 6100 Darmstadt, Germany
e-mail: neuhold@darmstadt.gmd.de, turau@darmstadt.gmd.de

1 Overview

At the Integrated Publication and Information Systems Institute (IPSI) of the GMD (Gesellschaft für Mathematik und Datenverarbeitung) database research is focused towards distributed database management systems to support the integration of heterogeneous multi-media information bases needed in an integrated publishing environment. The objective is to investigate advanced object-oriented and active data modelling concepts together with the principles of distributed data stores and data management. The unifying basis for the database research is the object-oriented data model VML developed in the project VODAK over the last four years. The data model is based on recursively defined meta classes, classes and instance hierarchies paired with a strict separation of structural and operational definitions in a polymorphic type system.

This report describes the highlights of the work on the VODAK database system, the research efforts in heterogeneous database integration and the research plans of the multimedia database project as well as applications of our system in different environments such as hypertext and office automation.

2 The VODAK Project

Participants: W. Klas, W. Chen, K. Drosten, H. Eisner, G. Fischer, H. Jadwiszczok, P. Muth, E. Neuhold, T. Rakow, V. Turau.

Research in the framework of VODAK focuses on an *extensible* data model and database programming language, and an advanced transaction model. Applications have been developed on the basis of the first prototype, some of them outside of IPSI (for example at the New Jersey Institute of Technology) in order to evaluate the concepts and to get feedback for improving the functionality of the system.

In the following we briefly describe the main activities and contributions of the VODAK group.

2.1 The VODAK Data Model Language VML

Usually database models lack mechanisms for extending them with additional modelling primitives. This limitation does not allow the adaptation of the models for specific application needs, e.g. database integration, multimedia document handling, hypertext modelling, etc.

The VODAK Model Language VML [K92a] homogeneously integrates the concept of *metaclasses* [K90] and the *separation of types and classes* [NGYT90] with other object-oriented concepts such as properties, methods, inheritance, and object identity. Complex nested data structures can be defined using the set, array, tuple, and dictionary type constructors. VML supports its own programming language for implementing methods, specifying transactions and an ad hoc query language.

In VML classes are used to organize a set of objects corresponding to real world entities and relationships between them. Object types define the structure of objects and the operations defined on these structures. They are associated with classes in order to determine the structure and behavior of the class' instances. Metaclasses are first class objects whose instances are classes. Metaclasses are associated with three object types: an (optional) *own-type* extending their own behavior, an *instance-type* specifying the behavior of their instances (which are classes), and an *instance-instance-type* specifying the behavior of the instances of their instances. Metaclasses can be organized in an instantiation hierarchy of arbitrary depth.

The concepts of VML can be used to adapt the model to specific application needs. For example, one can extend the data model to provide the general modelling concepts *NODE* and *LINK* as used in many hypertext systems by defining appropriate metaclasses describing the common structure and behavior of nodes and links, e.g. the creation date, the author, the navigation through a network, and some consistency constraints. This is done at the meta layer of a database schema and is independent

of any specific kind of nodes or links used in an application. Application specific nodes like *DATUM* or *CLAIM*, and application specific links like *SO* or *SINCE* (as used in argumentation networks) are defined at the meta-application layer of the database schema on the basis of the general concepts *NODE* and *LINK*. The classes and relationships of a specific hyperbase schema can be defined as instances of these metaclasses.

This approach leads to an open, adaptable data model which provides for the specification of additional modelling primitives at a meta layer of the database schema. The concept of metaclasses and the separation of classes and types allow to determine the structure and behavior of objects and the individual inheritance behavior via semantic relationships between arbitrary objects already at the meta layer independently from the specifications at the application layer for the application classes.

Applications developed on the basis of the first prototype include a multimedia application which incorporates a video laser disc providing all the database operations needed to view video on a screen [K92b], a database for argumentative networks (hypertext network according to the semantics of Toulmin schemas) for an authoring environment, and an example for the integration of an external database.

2.2 The VODAK Transaction Model

An essential system-internal function of a database management system is transaction management. In general, transaction management allows different users to access the same data concurrently in a consistent way and makes system failures, even crashes, as much as possible transparent to the users.

In VODAK, we focus on two specific problems of transaction management.

- Operations to read and edit (hyper)documents are typically complex, interactive and of long duration. A high degree of concurrency is required to reduce the number and length of times a transaction is blocked.
- A publication environment has to handle existing database systems for using and modifying remote information and documents. Transaction managers of existing systems, i.e. concurrency control and recovery, have to be integrated in a transparent way utilizing the functionality of existing managers.

Our transaction model is based on open nested transactions [RGN90, WHBM90, MRKN91]. Compared to conventional flat transactions, nested transactions allow more concurrency and are more flexible for recovery. A nested transaction is a tree-like structure, dynamically built up by the call of subtransactions until a bottom implementation level is encountered.

There are two types of nested transactions. Closed nested transactions provide a high degree of concurrency by allowing the concurrent execution of subtransactions. Furthermore, they allow to abort subtransactions independently of the calling transaction. Open nested transactions utilize the semantics of operations, i.e. a commutativity relation is used to define conflicts between operations. By considering the nesting structure, conflicts of subtransactions can be ignored if the calling (sub-) transactions commute and the called subtransactions are executed in a consistent way (i.e. serializably). For example, two increment operations on a counter commute whereas the corresponding update operations on a page conflict. After the update subtransactions are committed, their conflict can be ignored and the calling transactions can be executed concurrently.

We extended the open nested model from a fixed calling hierarchy of operations in a layered system (multi-level transactions) to an arbitrary calling hierarchy of operations in an object-oriented system [RGN90]. Commutativity of operations is applied to system defined VODAK methods, and to methods of user defined object types. For the latter we developed a framework to specify commutativity and inverse operations in VML [K92a].

We also developed an integration model for existing transaction managers. The open nested transaction model has been extended to support distributed transaction execution in an heterogeneous environment. The model provides ACID properties for global transactions if all existing transaction managers also provide these properties [MRKN91]. No assumptions about protocols are made. Subtransactions of existing systems constitute the leafs of our transaction tree. Again, we utilize the semantics of these transactions in order to achieve a high degree of concurrency. Recovery is based on inverse operations, compensating the changes of transactions which have to be aborted. Inverse operations are also used to ensure atomic commitment. Unilateral aborts as well as system crashes can be handled [MR91, MKN92].

For the sake of simplicity, in the current framework we do not allow the existing systems to execute local transactions that are not part of a global transaction.

2.3 The VODAK Prototype

The system architecture consists of a central database environment and several external database

environments to which the user wants to have integrated access. Each of these environments consists of an object manager, a message handler, a transaction manager, and a communication manager. In addition to these components an external database environment includes a database interface module which realizes the access to an external database system.

The DBMS components are currently built on top of DAMOKLES [DGL87]. The object manager handles terminal instances, classes, metaclasses, methods, primitive data types and type constructors (set, tuple, dictionary, array), and the mapping to/from DAMOKLES objects. The message handler executes messages sent to objects either locally in the central database environment or on a remote site running an external database environment depending on where the receiver object is located. The transaction manager provides services to execute methods according to the open nested transaction model. The communication manager handles the message exchange between the central database environment and the external database environments.

In addition to these components VODAK includes a syntax-driven VML editor and a VML compiler which translates VML schemas to DAMOKLES schemas, VML methods to C++ functions, and which generates appropriate objects in the VODAK data dictionary.

A first version of a C++ based prototype of VODAK is available for Sun Sparc Stations under certain conditions. It implements all the features specified in [K92a] including e.g. metaclasses, transactions, and remote message execution.

References

[DGL87] K.R. Dittrich, W. Gotthard, P.C. Lockemann: *DAMOKLES - The Database System for the UNIBASE Software Engineering Environment.* IEEE Database Engineering, Vol. 10, No. 1, 1987.

[K90] W. Klas: *A Metaclass System for Open Object-Oriented Data Models.* Doctoral Thesis, Technical University of Vienna, 1990.

[KNS90] W. Klas, E. Neuhold, M. Schrefl: *Using an Object-Oriented Approach to Model Multimedia Data.* Computer Communications, Special Issue on Multimedia Systems, Vol. 13, No. 4, 1990.

[K92a] W. Klas et al.: *VML - The VODAK Model Language Version 2.0.* Working paper, GMD, 1992.

[K92b] W. Klas: *Tailoring an Object-Oriented Database System to Integrate External Multimedia Devices* Workshop Heterogeneous Databases & Semantic Interoperability, Boulder, 1992.

[MKN92] P. Muth, W. Klas, E. Neuhold: *How to Handle Global Transactions in Heterogeneous Database Systems.* to be published in: Proc. 2nd RIDE-TQ. IEEE, Los Almitos, 1992.

[MR91] P. Muth, T. Rakow: *Atomic Commitment for Integrated Database Systems.* Proc. of IEEE 7th Int. Conf. on Data Engineering, Kobe 1991.

[MRKN91] P. Muth, T. Rakow, W. Klas, E. Neuhold: *A Transaction Model for an Open Publication Environment.* In: A. Elmagarmid (Ed.): Database Transaction Models for Advanced Applications. Morgan Kaufmann Publishers, San Mateo, 1991.

[NGYT90] E. Neuhold, J. Geller, Y. Perl, V. Turau: *A Theoretical Underlying Dual Model for Knowledge-Based Systems.* Proc. First Int. Conf. on Systems Integration, Morristown, 1990.

[RGN90] T. Rakow, J. Gu, E. Neuhold: *Serializability in Object-Oriented Database Systems.* Proc. 6th Int. Conf. on Data Eng., Los Angeles, 1990.

[TD92] V. Turau, H. Dûchene: *Equality Testing for Complex Objects based on Hashing.* To appear in: Data and Knowledge Enginnering, 1992.

[WHBM90] G. Weikum, C. Hasse, P. Broessler, P. Muth: *Multi Level Recovery.* Proc. of 9th Symp. on Prin. of Database Systems, Nashville, 1990.

3 The KODIM Project

Participants: P. Fankhauser, R. Busse, M. Kaul, E. Neuhold, M. Oheimer, Y. Xu.

KODIM develops methodologies and tools for enriching and integrating existing databases. The databases are *heterogeneous*, i.e. they model overlapping and related information with differences in naming, scaling, granularity, structure, and semantics, and are *autonomous*, i.e. they cannot be modified for the purpose of integration, and their schemas can evolve independently. Their integration involves the *enrichment* of data, the *recognition* of related data, the *resolution* of inconsistencies in their representation and their *merge* by means of integrated views.

For the *recognition* of semantically related data we employ background knowledge in the form of a fuzzy thesaurus [FKN91]. With this approach, the likelihood that two classes meaningfully share data is determined on the basis of their semantically relevant attributes.

To resolve structural inconsistencies we utilize rule based heuristics [SN88a]. Rules are used to propagate messages, which cannot be dealt with by a given class, to semantically related classes, and to combine answers received from more than one class.

To merge overlapping data we utilize high level operators for generalization, specialization, grouping, and aggregation, which are modelled by the metaclass facility of the VODAK datamodel [SN88b].

On this basis we have implemented *ViewSystem* [KDN90], which allows for the definition of integrated classes by means of metaclasses, and supports query decomposition and view materialization as two strategies to instantiate the integrated classes.

Currently, we extend the rules of [NS88] to cover more kinds of inconsistencies, and to assist also the merging of complex data on the basis of simple correspondence assertions given for their constituents. These rules are being implemented in Prolog. Specific emphasis is given to inference strategies which do not require that all correspondence assertions are available, but prompt the user for missing or uncertain assertions [FN92]. Furthermore, we design rules for integrating methods consistently with the attributes and methods they utilize. This rule base design intelligence is the basis of a graphical user interfaces for defining and integrating schemata.

The project cooperates also with the New Jersey Institute of Technology, and indirectly with Bellcore, to develop further integration mechanisms and to experiment with heterogeneous database environments [GPN91a, GPN91b].

References

[FKN91] P. Fankhauser, M. Kracker, E. Neuhold: *Semantic vs. Structural Resemblance of Classes.* SIGMOD RECORD Vol. 20, No. 4, 1991.

[FN92] P. Fankhauser, E. Neuhold: *Incompleteness and Explanation in Dynamic Schema Integration.* Workshop Heterogeneous Databases & Semantic Interoperability, 1992.

[GPN91a] J. Geller, Y. Perl, E. Neuhold: *Structural Schema Integration in Heterogeneous Multi-Database Systems using the Dual Model.* Proc. Int. Workshop on Interoperability in Multibase Systems, Japan, 1991.

[GPN91b] J. Geller, Y. Perl, E. Neuhold: *Structure and Semantics in OODB Class Specifications.* SIGMOD RECORD Vol. 20, No. 4, 1991.

[GMPNS] J. Geller, A. Mehta, Y. Perl, E. Neuhold, A. Sheth: *Algorithms for Structural Schema Integration.* Research Report CIS-91-31, NJIT, 1991.

[KDN90] M. Kaul, K. Drosten, E. Neuhold: *ViewSystem: Integrating Heterogeneous Information Bases by Object-Oriented Views.* Proc. of the 6th Int. Conf. on Data Engineering, 1990.

[NS88] E. Neuhold, M. Schrefl: *Dynamic Derivation of Personalized Views.* Proc. 14th VLDB, 1988.

[SN88a] M. Schrefl, E. Neuhold: *A Knowledge Based Approach to Overcome Structural Differences in Object Oriented Database Integration.* in: The Role of Artificial Intelligence in Database & Information Systems, 1988.

[SN88b] M. Schrefl, E. Neuhold: *Object Class Definition by Generalization Using Upward Inheritance.* Proc. of the 4th Int. Conf. on Data Engineering, 1988.

4 The Multimedia Project

Participants: V. Turau, T. Rakow.

Many types of information which were previously only represented in analog form are now also available in digitized form. The most important examples are audio and video. This offers the possibility that all information can be processed by a computer. Thereby the multiplicity of ways data can be utilized, manipulated and displayed is enormously increased. The aim of the multimedia project is to provide databases with the capability of storing, managing and retrieving information on individual storage media, managing interrelationships between the information on different media and means to exploit these media for presentation purposes.

At the conceptual level a uniform and homogeneous data model based on the VODAK system will be provided to capture the structural and behavioral aspects of the objects.

In a multimedia system there will be several physical devices used to store data: magnetic or optical disk, CD-ROM, video disk, DAT etc. Therefore, extensibility is a very important goal of the project, since invariably new storage devices will be on the market in the near future, some of them heterogeneous in the sense described in section 3.

At the external level the presentation of the diverse media to the users will be handled by an appropriate user interface allowing for easy manipulation for these new types of information. A suitable query language is under development to retrieve the desired multimedia information, present the results of queries and manage the results for referencing in later queries. This language should support queries against the content of multiple multimedia systems. In many media types this can only be achieved by attaching the appropriate annotations to the data and pose queries against these annotations. Therefore, powerful browsing facilities are necessary to select the desired object from a set of qualifying objects. In the future content analysis/generation features such as image-, video- and speech recognition/generation will have to be added to increase the manipulative power of the system.

The project will cooperate closely with HP Labs, Palo Alto, were a small initial prototype of a multimedia database has been developed by one of our members using the Pegasus prototype [T91].

References

[T91] V. Turau: *Multimedia Database Systems.* HP Report, 1992.

5 Applications of the Vodak Data Model

5.1 Hyperbase

Participants: H. Schütt, J. Haake, W. Schuler, N. Streitz.

Hypermedia systems allow for the manipulation of pieces of information (*NODE*) together with their relationships (*LINK*). Over the last few years, many prototype systems have been developed, and some have successfully entered the market place. Common to most of them is their lack of adequate support for collaborative work, but also their lack of consistency, security and reliability control. They usually store all data in files and do not provide any means for transaction management.

The division *Cooperative Hypermedia Systems* is concerned with the development of SEPIA, an authoring system for hypermedia documents [SHT89, THH91]. Within that system, an application-independent storage layer for hypermedia objects (i.e. a *hypermedia engine* called *HyperBase*, [HS90, SS90]) has been designed and implemented.

Using a DBMS within a hypermedia engine allows a hypermedia system to share data among several users, to protect these data through appropriate transaction management, and to access data through indexes to be fast enough for interactive use.

The implementation of the first *HyperBase* prototype utilizes modelling concepts from the VODAK data model but still uses a commercial relational DBMS for the physical storage of data. The experience has shown that the Hyperbase data model is adequate for different user-interfaces of a hypermedia system. The chosen fine-grain transaction and locking schema on the basis of the closed nested transactions of the underlying database proved useful for that kind of system, and its performance is sufficient for short transactions in interactive applications.

Future research topics in the project are: Development of a HyperText Query Language (HTQL) on the basis of the VODAK query language and the VODAK data model. It will provide long and cooperative transactions, use multiple, distributed servers for a hypermedia engine, and integrate data from external databases into the hypermedia system. A second prototype will then be implemented based on VODAK and VML, which already provides the basis for most of the needed functionality.

5.2 Structured Document Base SDB

Participants: C. Hüser, A. Weber.

SDB is a prototypical database application for the maintenance of structured documents. SDB provides persistent storage for structured documents, offers an interface for the manipulation of structured documents and a query language for declarative access to structured documents. SDB supports an import/export interface conforming to the SGML (Standard Generalized Markup Language) standard. Thus, the SDB builds a powerful basis for the realization of distributed publishing applications, such as the Individualized Electronic Newspaper (IEN) developed in the Race Project 1075 TELEPUBLISHING [H91].

The application-independent architecture of the SDB incorporates an object-oriented Document Access Interface (DAI) for rapid prototyping and for integrating existing tools. The integrated concept for the management of structured documents frees tools from dealing with questions of storage management and enables data sharing. Conventional database systems cannot provide sufficient functionality to support the needs of applications in the publication field, therefore the object-oriented database system developed in VODAK is used; especially its extendibility features through the meta class system allows the modelling of structured documents.

SDB supports the management of pools of structured documents. It implements a subset of the basic SGML Document Standard.

A pool is an organizational unit that provides all information for document access and holds a collection of Document Type Definitions (DTDs) and document instances that conform to one of the DTDs stored in the pool. A DTD contains a content structure specification for a class of documents. It guides the content acquisition and structuring of the document content. A basic feature is the definition of elements to specify the content structure. Following object-oriented terminology an element declaration is an object class description. The attribute declarations of an element define properties of the object. The content model of the element declaration defines the allowed subelements. Starting from the root element declaration in the DTD the content models of all element declarations specify the logical structure of a document and correspond to a context-free grammar. The logical structure of a specific document instance is represented as a parse tree.

DAI provides the management mechanisms for pools, but also the functionality for developing applications for different uses of document structures and for different levels of access. Navigation and retrieval methods for enduser applications such as browsers, and retrieval interfaces that generate different views of the selected contents, and manipulation methods for structure or hypertext editors are provided.

5.3 User Interfaces

Participants: M. Kracker, A. Müller, E. Neuhold.
Object-oriented database models like VML with inheritance and type constructors allow the easy construction of complex schemata, but they also put a heavy burden on anyone working with such databases. Because queries have to reflect the schema or predefined views of the respective databases, a searcher not truly familiar with the conceptual organization of the data might introduce mismatches in naming, structure or scaling, which prohibits the normal execution of the query.

To temper the consequences of mismatches, where a database construct, e.g. an object or a method, is named differently by the database administrator and the database user, we have developed a query formulation support system (for an early version see [KN89]). It handles parts of the query which do not directly denote database constructs by working with the data dictionary and the schema description it contains. It suggests the "closest corresponding" database constructs for the unidentified query components and thus ultimately allows the execution of such queries.

The measure of similarity does not only consider the degree of structural correspondence, but also the semantics, which is expressed by the names given to object classes or methods. Semantic similarity between two names are derived from a fuzzy associative network, which relates the concepts of a given discourse domain and which can be applied to multiple databases of that domain. Because flat modelling of the domain is sufficient and we wanted to keep the construction effort of the concept network low, we introduced only a few types of relationships: generalization/specialization, similarity and opposition. A value between 0 and 1 is assigned to each relationship to express its strength. For example different degrees of similarity, like synonymy or relatedness can be expressed easily by these fuzzy relationships. Utilizing the transitivity of the relations, we can automatically derive further relations and increase the density of the associative net. The calculus of similarity is defined on a meta level, which controls the inference by applying the appropriate transitivity operators.

The fuzzy concept network, the inference mechanism and the specification of the calculus of similarity are described in detail in [K91] and [K92]. In addition, we have developed a prototype of the query formulation support system for an bibliographic frame-like database and we use it for experimentation.

A user centered query interface to a relational database containing a download of EC-CORDIS (the research and development information system of the European Community) has been developed ([T92]). To help the searchers find alternative terms while formulating a free text search, we again employ a fuzzy concept net. Here the support of term associations is given on the instance level, as opposed to the application described earlier, where database constructs on the schema level are processed. Our approach will be extended to the direct access of remote hosts and their dynamic partial download during query time. We are also investigating the application of our approach to VML queries [K91]. Utilizing fuzzy concept networks for database integration has been discussed in [FKN91] (see section 3).

References

[HS90] J. Haake, H. Schütt: *Eine Systemarchitektur für ein wissensbasiertes Hypertext-Autorensystem.* In: P. A. Gloor, N. A. Streitz, Editors, *Hypertext und Hypermedia: Von theoretischen Konzepten zu praktischen Anwendungen*, Springer Verlag, Inf. Fb. 249, 1990. In German.

[H91] C. Hüser: *Report on a prototypical interface for structured documents and its application to the IEN scenario.* RACE Project Telepublishing 5.1.1-1, 1991.

[KN89] M. Kracker, E. Neuhold: *Schema Independent Query Formulation* Proc. of the 8th Int. Conf. on Entity-Relationship Approach, Toronto, 1989.

[K91] M. Kracker: *Fuzzy Associative Concept Knowledge for Supporting the Formulation of Database Queries.* PhD-Thesis, Technical University Vienna, Austria, 1991. In German.

[K92] M. Kracker: *A Fuzzy Concept Network Model and Its Applications.* Proc. of the FUZZ-IEEE'92, IEEE Int. Conf. on Fuzzy Systems; San Diego, 1992.

[SHT89] N. A. Streitz, J. Hannemann, M. Thüring: *From Ideas and Arguments to Hyperdocuments: Travelling through Activity Spaces.* In: Proc. Hypertext '89.

[SS90] H. Schütt, N. Streitz: *HyperBase: A Hypermedia Engine Based on a Relational Database Management System.* In: Proc. ECHT '90.

[T92] U. Thiel et al.: *Towards a User-Centered Interface for Information Retrieval: The MERIT System.* Submitted for publication, 1992.

[THH91] M. Thüring, J. Haake, J. Hannemann: *What's ELIZA doing in the Chinese Room - Incoherent Hyperdocuments and how to Avoid them.* In: Proc. Hypertext '91.

[W91] A. Weber: *Publishing Tools Need Both: State-Oriented and Task-Oriented Version Support.* 5th Int. Comp. Softw. and Appl. Japan, Conf. 1991.