Hyperdatabases

Hans-Jörg Schek, Klemens Böhm, Torsten Grabs,
Uwe Röhm, Heiko Schuldt, Roger Weber,
Swiss Federal Institute of Technology, Zurich Switzerland
{schek, boehm, grabs, roehm, schuldt, weber}@inf.ethz.ch

Abstract

When relational database systems have been introduced twenty years ago, they were an infrastructure and main platform for application development. With today's information systems, the database system is a storage manager, far away from the applications. Our vision is that hyperdatabases become available that move up and extend database concepts to a higher level, closer to the applications. A hyperdatabase manages distributed objects and software components as well as workflows, in analogy to a database system that manages data and transactions. In short, hyperdatabases, also called "higher order databases", will provide "higher order data independence", e.g., immunity of applications against changes in the implementation of components and workload transparency. They will be the infrastructure for distributed information systems engineering of the future, and they are an abstraction from the host of current infrastructures and middleware technology.
This article will elaborate on this vision. We will outline concrete projects at ETHZ such as PowerDB, a database cluster project. We show how an efficient document engine can be built on top of a database cluster. A further project studies transactional process management as a layer on top of database transactions. Image similarity and multimedia components is another project where a hyperdatabase coordinates specialized components such as feature extraction and indexing services in a distributed environment.

1 Introduction and Background

When relational database systems have been introduced twenty years ago, they have been considered as infrastructure and main platform for development of data-intensive applications. Standard database textbooks list this as a main motivation in the introduction. Data independence was considered to be a breakthrough: programmers were freed from low-level details, e.g., how to access shared data efficiently and correctly, given concurrent access. But by now, the prerequisites for application development have changed dramatically. For instance, communication has become fairly cheap, and the internet dominates modern information infrastructures. Consequently, the role of database concepts must be re-visited and newly determined.
Let us consider current 'hot topics', e.g., the management of software components and application services, distributed client/middleware/server computing, application frameworks, enterprise resource planning systems, XML, and e-commerce. Undoubtedly, the database system plays an important role. However, it has degenerated to a storage manager to a large extent, far away from the applications.
More and more researchers are making these observations and start to question the role of databases for future distributed and WWW information systems engineering. They re-orient their research to make well-founded and established database concepts more applicable, as the following episodes show.
Carey, Hellerstein and Stonebraker [Regr] observe that all current databases have been designed with the technology of twenty years ago. They state that due to a three-tier architecture, data are at the bottom and application code is away from data in the middle tier. They also state that databases are "bloated" by object-relational features, by stored procedures and triggers, and by warehouse features. They conclude that we should rethink everything.
In a keynote speech Brodie [Bro99] states that "the database era nears its end" because of their inability to cope with the vast increases in data and transaction volumes. Heterogeneity hinders interoperability and accessibility. Architectural complexity is another issue in view of the many engine and repository types with ad-hoc solutions for warehousing and mining.

The VLDB Endowment in 1998 has established a future directions group[1]. To bring application development back into the fold of database conferences, this group proposed to distinguish between the two main directions: (1) core database technology, and (2) infrastructure for information system development. Accordingly, the future program committees will be re-organized [VLDB]. (For more details and other interesting observations on the evolution of database research the reader is referred to documents such as [Si+96].)

Even though the general awareness of the problem slowly proliferates, the database group at ETH Zurich has been involved not only in core database technology, but also in the cooperation between databases and specialized application services for more than ten years. "Cooperative Service Management for Open Systems" (COSMOS) is our research framework since 1990 [SSW90].

Taking all these observations together, database groups, while keeping competency in core database technology, start moving up to higher levels where workflow and distributed component and object management takes place. Their results will influence the next-generation platform for information-system development. The work on hyperdatabases is the contribution of the ETH Zurich database group to this vision.

In the following, we elaborate on this vision and describe concrete projects. The presentation is similar to [ScW00], but it is more complete with respect to project descriptions. [AHST97] is also related and puts more emphasis on workflow and process management.

2 The Hyperdatabase Vision

2.1 What is a Hyperdatabase?

First rough definitions are: A Hyperdatabase (HDB) is a database over databases. An HDB administers objects that are composed of objects and transactions that are composed of transactions. Like hyper-matrices in numerical analysis are matrices whose elements are matrices an HDB is a database, the primitives of which are again databases. Therefore instead of talking about hyperdatabases we may also call them "higher order databases". We will use both terms synonymously. In a more general setting we say that an HDB administers distributed components in a networked environment and provides a kind of higher order "data independence". Remember that data independence was a major breakthrough when relational databases were propagated. Now we must strive for immunity of application programs not only against changes in storage and access structure, but also – and this is the point here – against changes in location, implementation, workload, the number of replica of software components and their services.

What is the difference between a DB and an HDB? In a nutshell we say: a DB is a platform for clients concurrently accessing shared data. We need data definition, data manipulation, and transactions at the interface. The DB under the cover performs query optimization, correctness for parallel access, recovery, persistency, load balancing, admission control, availability.

Similarly, an HDB is a platform for clients, concurrently accessing shared application services. As opposed to shared data in a DB, in an HDB we have shared components and services. At the interface of an HDB we need component and service definition and description, service customization, transactional processes encompassing multiple service invocations. The HDB, under the cover, performs optimization of client requests, routing, scheduling, parallelization, correctness of concurrent accesses, flexible failure treatment, providing guaranteed termination, availability, flexible recovery, and scalability.

2.2 Hyperdatabases and Middleware

The general architecture where HDB concepts are used is shown in Fig. 1. Clients (white boxes) invoke application services (black boxes) on server components that are coordinated (large rectangle). One application invocation in turn may use several other coordinated components and invoke one or several lower-level application services there. This continues along the invocation hierarchy until we finally reach leaf nodes. The leaf nodes perform data calls or specialized computations (gray boxes). Note that coordinator nodes play the typical middleware dual role: they are servers for clients and clients to other servers. The repeated use of coordinators adds complexity because at any server node we may have additional clients also accessing our distributed system concurrently to others coming from higher level coordinators. Seen from this perspective the HDB concept helps us to manage n-tier architectures that exist and will further be our realistic information system infrastructure.

Many products or standards and proposed architectures provide already the one or the other functionality of an HDB, such as TP-monitors now evolving to object monitors [BoK99], CORBA [CORBA], Enterprise Resource Planning systems, federated databases, and data warehouses.

All of them provide the one or the other desirable feature of an HDB. The HDB in turn provides a unified view of

[1] Members were Agrawal, Brodie, Carey, Dayal, Gray, Ioannidis, Schek, Wang and Widom.

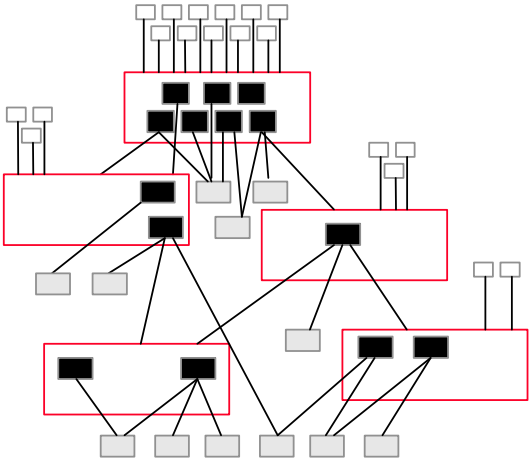

Fig. 1: Repeated usage of an HDB in a distributed n-tier architecture. Clients invoke application servers at several coordinators.

all these approaches, incorporating the essential principles. It incorporates database concepts, but at a higher level of abstraction than with records or tuples. This should allow for a better understanding and for convenient implementations. This is the objective of the HDB project. In the following we discuss some examples of current architectures and sketch the opportunities for improvement.

2.3 TP Monitors

A TP-Monitor is - in a narrow definition - an operating system for transaction processing. More generally it is infrastructure for developing and running applications, services, and components in a three-tier architecture. While TP-Monitors are definitely helpful, and they are used in many large-scale implementations, we see some room for extensions: an HDB would add design tools for specification of distributed applications and automatic generation of program code, as extended workflow systems do [AHST97]. Further, an HDB would add a real transaction layer above DB-transactions and transactional process management with failure treatment, availability, guaranteed termination and "semantic" correctness. In current TP-monitors or object monitors, all components are “transactional". This means that they are forced to the 2PC protocol. Coordination of all subtransactions is performed at one level of abstraction only, namely the data level. Finally, an HDB would add optimal routing to components and clever "semantic" replication of whole components.

2.4 Enterprise Resource Planning, SAP

As a next example we consider the architecture of the SAP system, a good representative of ERP software. Clearly we observe the traditional DBMS at the storage level. All application code is at the middle tier in so-called application servers and presentation stays at the utmost level. The architecture is the one of an HDB. But a closer inspection of the role of the middle layer and of the mapping to the DB layer in a SAP system makes clear that a real HDB layer would add the following:

Most importantly, there should be a real transaction layer on application objects above DB-transactions and transactional process management with semantic recovery and semantic correctness as well as the possibility of alternative executions in case of failures and guaranteed termination. In the current system there is a separate transaction layer on top of DB transactions called logical unit of work which is in the sense of an HDB. However application programmers must aquire locks and so they are responsible for correctness in case of parallel access and in case of failures. Also consistency of application server buffers in case of updates is not provided automatically.

Second, an HDB would add flexible mapping to many database storage managers. In the current system there is one big database only, hindering scalability. If more than one storage component is allowed, optimal decomposition and routing of client requests to storage managers can take place and a clever replication of whole storage components based on usage patterns can be achieved.

Third, application server caches would be kept consistent and would not be under the responsibility of the application programmer, as is the case with the current version of SAP. An HDB would automatically optimize various caches and decide which objects to cache.

3 Hyperdatabase Projects

In the following we summarize some of our projects at ETH Zurich in order to give a more concrete explanation on what we mean by an HDB approach.

3.1 Composite Transactions and Transactional Process Management

Foundation. We have studied the problem of ensuring correctness of concurrent executions in architectures like the one in Figure 1. A coordinated server component, called coordinator in the following, contains a full transaction manager for the client transactions encompassing invocations of several application services.

Every coordinator in the composite system performs transaction management on its own, ensuring (local) correctness and (local) recovery. The problem is how global correctness and global recovery is ensured, given that each coordinator locally works correct. In other words we must know what additional ordering restrictions a "caller" coordinator must impose and hand over to a "called" coordinator. What handshaking between coordinators is necessary in order to correctly control concurrency in composite systems as shown in Fig. 1. We have extensively studied this problem in the past from a foundational point of view (e.g. in [Wei91, WeS92, SWY93, Alo+97, Alo+99a, Alo+99b]), we have studied practical protocols [SWS91, SSW95] and performed several evaluations [KaS96, RNS96].

Transactional Process Management. In the transactional process management project, we go beyond the conventional transaction model by grouping single transactions into entities with higher level semantics, called *transactional processes*. These processes encompass flow of control as one of their basic semantic elements. Aside of constraints for regular executions, it is also possible to specify alternative executions, which can be applied in case of failures. The steps that have to be executed within processes are invocations of arbitrary complex transactions provided as services by the systems of the next lower level. These transactions may differ in terms of their termination behavior, which indicates whether they can be compensated and whether they lead to success after repeated invocation (retriability) [MRSK92, ZNBB94]. Given the termination behavior of single transaction invocations and the specification of control flow and alternatives, single processes can be proven correct. This is captured by the notion of *guaranteed termination,* which generalizes the traditional notion of transaction atomicity. Having this inherent property, a process, once invoked, terminates in a well-defined state by correctly executing one of possibly several alternatives. Abort is a special option, forward recovery (e.g., partial backward recovery combined with alternative executions) another. The guaranteed termination will also be ensured if several transactional processes are executed concurrently. Their more complex structure implies more complex dependencies compared with traditional transactions and has to be considered when transactional processes are scheduled [SAS99], thereby extending previous work on concurrency control and recovery [Alo+97, SWY93]. Similar approaches on combining transaction management and process execution can be found, for instance, in [WR92,CD96]. Since we do not assume all applications managed by coordinators as depicted in Figure 1 to be pure database systems, the transactional properties required for single invocations may not always be present. Given some basic requirements we have analyzed in [SSA99], a transactional coordination agent (TCA) is plugged to the application, thereby extending its functionality by adding transactional properties to service invocations (e.g., [NSSW94]). When considering, for instance, coordination processes in multimedia information systems (which we discuss subsequently), these agents are extended to capture the workload of single components, which can be exploited for load-balancing purposes.

Work on transactions and transactional processes is a foundation and a basis for HDB transaction implementations. These principles have been applied, for instance, in the context of payment interactions in electronic commerce which are mapped to processes, thereby making use of the execution guarantees provided by a transactional process manager [SPS00].

3.2 PowerDB

In the PowerDB project we explore an HDB consisting of a homogeneous set of component databases in a PC cluster. In every component, we have a complete DBMS with its data. Clients access data via the coordinator, i.e. via the HDB (fig. 2). We explore protocols for high-level transaction management under special consideration of replication. Replication of complete databases contributes to considerable speed-ups in case of read transactions. Due to the second layer transaction management we avoid a 2PC and avoid synchronous updates [GBS99,GBS00]. In addition, query routing aims at detecting components that have the shortest response time due to queries that have been processed before [RBS00]. Replication can be full or be restricted to certain parts of the database. We investigate methods that dynamically allow to add more components to the cluster. The subprojects "Routing of OLAP requests in a DB Cluster" and "Document

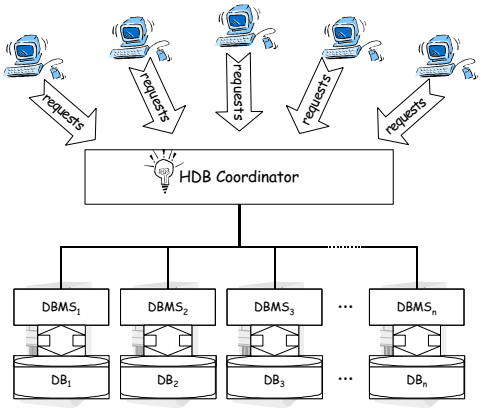

Fig. 2: The PowerDB Architecture

Management on a DB Cluster" are described below in more detail.

Routing of OLAP requests in a DB Cluster. Online Analytical Processing queries (OLAP queries) refer to data warehouses i.e. to large collections of data, extracted and accumulated from operational databases. Despite their complexity, users want OLAP queries to be evaluated fast.

The key to good retrieval performance is an appropriate physical data organization combined with query routing. In [RBS00], we have compared full replication and a hybrid placement scheme combining partial replication with partitioning. Both approaches replicate at least some data over all nodes. Hence, there are several components in the general case that can evaluate a query. The Query Routing component decides at the HDB level which component is best suited to evaluate the query. We have built a prototype system for PowerDB including – among other features – a second layer transaction management, a routing component providing different routing strategies and a parallel distributed query processor (based on [RB99]). Using this prototype, we have done performance evaluations with the TPC-R benchmark comparing the different routing strategies: *simple*, round-robin-kind-of strategies like *random, first-come-first-server* or *Balance-the-Number-of-Queries* [CLL85], and a more sophisticated *affinity-based* routing strategy which assigns queries accessing the same data to the same component [YCDI87, RBS00]. It turned out that the PowerDB architecture in general offers a linear speedup of mean response time with increasing number of components (cf. Figure 3). The hybrid data placement scheme proved superior to full replication, showing an even more than linear speedup due to caching effects: the single fragments of the partitioned relation are getting smaller with increasing cluster size, which notably improves cache hit rates per component. Besides, we observed that OLAP queries evaluated concurrently typically obstruct each other, leading to a certain performance penalty. For example, the mean response time when executing queries concurrently per node up to a multiprogramming level of

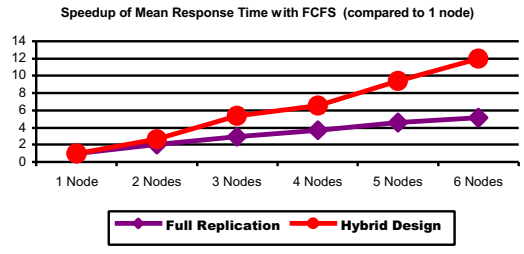

Fig. 3: Query Routing - Response Times

6 is 180% higher than with FCFS as shown in Figure 3. This effect can be avoided by using affinity-based query routing, which improves the mean response time as compared to FCFS-routing by 30 percent on average.

We are currently developing routing strategies, which keep track of the component's cache state and route queries to the node with the best-suited cache content.

Document Management on a DB Cluster. XML [W3C98] proliferates, notably for information exchange for e-commerce. E-commerce applications require an efficient access to large collections of XML-documents and the immediate and permanent availability of these documents after they have been submitted. But most of today's document management systems still assume that new documents are inserted during an off-line period of the system. This also holds for many current approaches to XML databases where load times for large data sets are still prohibitive [FlK99, ST+99]. The major drawback with this approach is that the querying user never operates on up-to-date data. [KaS96, KaR96, BMV96] address this problem with systems that allow for concurrent retrieval and insertion of documents. Both approaches are based on multi-level concurrency control [Wei91] to prevent from inconsistent schedules without sacrificing parallelism of lower level operations. But, in this previous work, the issues of scalability via a database cluster and conformity to the new document format XML have not been addressed. In our current investigations of document management, a higher order data object is an XML document. The HDB provides services for these higher order data objects, such as insertion, retrieval and deletion. Under the cover the HDB maps these services onto a cluster of databases running on off-the-shelf PCs (similar to [ink96] and [FG+97]). The HDB additionally provides transaction management for document services. This leverages conventional database technology to document databases.

The great advantage of a DB cluster is that it gives us the freedom to partition and replicate data and indexes and to allocate them to as many cluster components as necessary. Decomposition and parallelization of requests together with composite transactions implemented at the coordination and the service layer of the HDB ensure short query response times and further allow for concurrent update and retrieval requests. Fig. 4 shows the architecture of such an HDB.

We have implemented a prototypical document database for a specific type of documents (i.e., for newsgroup postings) [GBS99, GBS00]. A cluster of relational database systems stores the document texts and the inverted lists. Our experiments show that document insertions scale linearly, and retrieval operations slightly less than linear for increasing cluster sizes (c.f. Fig. 5).

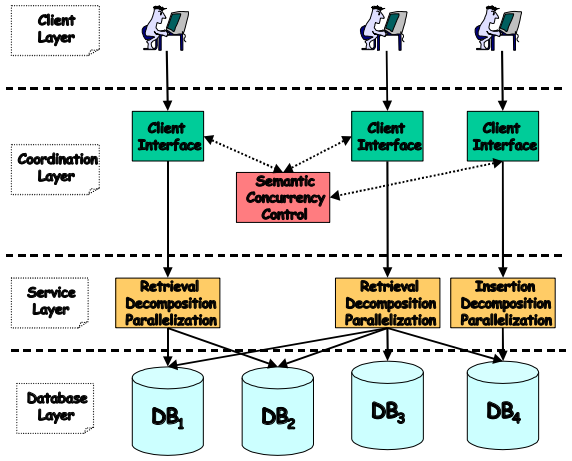

Fig. 4: Document Management Hyperdatabase

We observed that average response times are interactive for mixed workloads of 40 parallel insertion and retrieval streams already on a cluster with only 8 database nodes.

In extension to this prototype system, our current work focuses on documents with arbitrary structure and no restriction of the acceptable DTDs.

An essential design issue is to reduce the amount of centralized processing in a clustered environment. This is important when the cluster size goes into some hundreds of database nodes that have to be coordinated. Our approach takes this into account as document specific functionality is distributed among the components. Only a table with information for semantic locking is necessarily kept at a centralised coordinator node. Results in [BGRS00] are encouraging.

In our future work, we want to address a self-adapting hyperdatabase system. At the data object level, such a system automatically partitions, replicates and materializes data in order to process requests efficiently. At the service level, such a system replicates functionality among the coordinated nodes in order to meet the changing processing needs of its clients.

3.3 Image Search and Management Systems.

Information systems for image collections consist of many specialized software components such as image servers, image processing, feature extraction, and indexing components. In such a setting, given the possibly large number of components and the high workload imposed on them, location and workload transparency is of great value, and the HDB vision applies here.

In more detail, we use a PC cluster, which contains as many specialized software components installed as

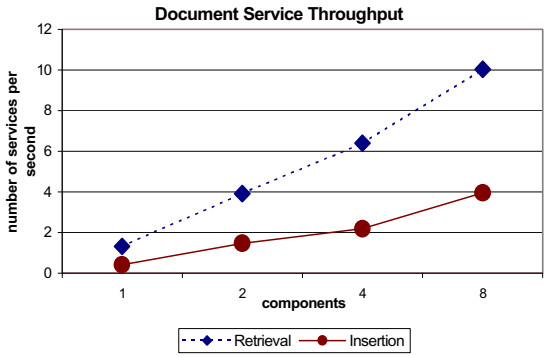

Fig. 5: Document Service Scalability

necessary and coordinate them by an HDB. For the coordination, we need descriptions of the software components including its actual load. Simple transactional processes for insertion, similarity search, and bulk load can run in parallel and the subtasks are "optimally" assigned to the components by the HDB. At any point in time new components can be added to the cluster in order to improve response times. Interactive similarity retrieval is based on the VA-File, a simple but efficient approximation of the inherently high-dimensional feature vectors [WSB98,WeB00]. In the following we summarize the infrastructure aspect and the parallelization necessary for interactive similarity search and relevance feedback.

Coordination of the Image System Component. In order to let the HDB coordinate the various components, not only static information about the components is necessary but also the dynamically changing state of every component at any point in time is required. For this purpose we have introduced *coordination agents* [SSA99,Web+99] that are plugged-into all components (fig.6). A coordination agent observes the status and actions of its component and initiates processes that make sure that all dependencies between components are properly maintained. Examples are processes for index maintenance, replication control, or consistency of metadata. The specification and execution of such processes are the main tasks of the HDB assisted by the agents. For instance, if a new image collection extends a repository, the agent of that repository initiates a **bulkload** process to extract the required features from the images and to insert these features into specialized index structures for similarity search.

The main problem of a bulkload process is the enormous cost of extracting features: e.g. for 100,000 images, this pre-processing step may last up to 50 days on a single machine. To speedup the extraction, the HDB deploys a large number of feature extraction components in parallel.

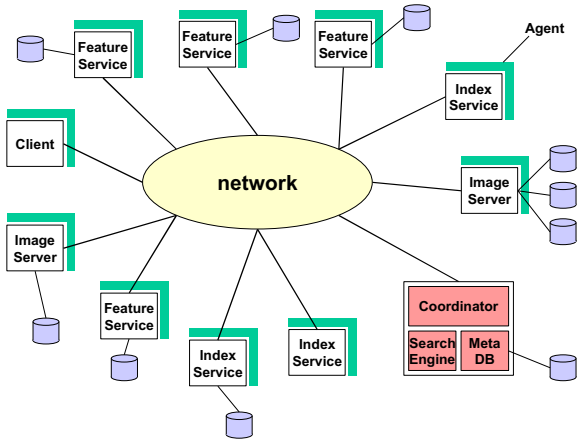

Fig. 6: Image System HDB

Each component works off a part of the images in the new collection. If the extraction of features would last too long, we can dynamically add new components to the system. After their registration, the HDB will account for these new components and route requests to them without the need of reconfiguring or restarting the system. Even processes that have started before adding a new component can benefit from the additional component. The same holds true if components retire from the system or fail: the coordination agents notify the HDB about this, that component is excluded from future processing, and its pending tasks are re-routed to other available components.

Load balancing is a further key aspect of the HDB in our image database system. In a bulkload process, for example, the pre-processing cost of an individual image depends on the size of the image and of the component that does the extraction (heterogeneous environment). In order to minimize the cost of the entire process, the HDB must balance the load over the available components as equal as possible. To this end, the coordination agents of the extraction services periodically inform the HDB about the current load of the service (in our case: the time required working off all tasks assigned to the service). Then, the HDB assigns a new extraction task to the service with the smallest current load, but only if its load is below a given threshold (in order not to assign all tasks to the components at once). With this policy, we can ensure that all components are (almost) equally loaded, and that new feature extraction components can be used to speedup up current ulkload processes. More details can be found in [[Boe+98, Web+99, WeS99, ScW00]. The same technique is used for other processes like similarity search as discussed in the following.

Parallel Similarity Search and Relevance Feedback.

The rationale behind our activities in the field of similarity search is to provide search mechanisms that are easy to use, and that yield results of high relevance to the user. Relevance and relevance feedback depends largely on the availability of many different feature types in order to enable the feedback mechanism to adapt to the (subjective) relevance judgement of a user. Consequently, similarity search over image collections means Nearest Neighbour search (NN-search) in feature spaces that are high-dimensional. The dimensionality typically is in the range of several hundreds if information on several feature types is combined. Thus, and this is the first issue, similarity search over large image collections requires considerable engineering effort in order to ensure interactive response times. In this situation, an HDB approach helps in the following way: it is able to coordinate as many feature extraction components and replica of indexing components as necessary, enabling high-level parallelization.

NN-search in high-dimensional feature spaces is provably linear in the number of data objects [WSB98]. In order to accelerate the unavoidable linear scan, we use bitstring approximations of feature vectors. This data structure is called the **VA-File** [WSB98, WeB00]. NN-search using the VA-File is as follows: a first phase explicitly inspects all bitstring approximations. This gives us candidates for the k nearest neighbours. The second phase checks the candidates and determines the k nearest neighbours. Note that the first phase is well-suited for parallelization. We can partition or replicate the VA File and distribute it over an arbitrary number of components.

However, a number of design decisions arise. A first issue is whether to replicate or to partition the approximation data. The tradeoff is flexibility vs. update costs. In more detail, flexibility means that the system can adequately react to changing behavior of components. For instance, the workload of a component may suddenly increase, or a component may become unavailable. The HDB vision is implemented as follows: the coordinator takes into account the current states of the components and finds an appropriate query evaluation strategy. This strategy remains transparent for the application on top. As in other projects described in this article, agents closely monitor the components and feed the coordinator with up-to-date information. Next to the workload, the coordinator knows about the cache state of the components by approximating it from the queries previously executed and takes it into account when assigning subqueries to components.

What are the performance characteristics of this implementation of the HDB vision? A central observation is that a relatively small number of components is already sufficient for interactive query-answering times. We believe that this is a major result, since much research on similarity search did not yield any comparable solutions.

Given such an efficient implementation of similarity search, the natural next step is to allow the user to formulate his information need by means of relevance feedback mechanisms. I.e., the system collects user feedback and interprets it, in order to refine the search in subsequent steps. The main idea is to map user feedback to a statement in a similarity query language and to evaluate the query. A number of such mappings have been proposed in literature, e.g., [Roc71,RHOM98]. Our solution has been to develop a framework that is extensible, and that allows for easy integration of the approaches that are around. A distinguished component implements the relevance feedback functionality. Its embedding into the similarity search process is relatively easy, thanks to our component-based approach. The spezialized VA-File component can easily be integrated and query answering times are again pleasingly low, even with complex relevance feedback queries,.

4 Conclusions

In a first part we have presented the general hyperdatabase vision. Its main objective is to provide a higher-level platform for distributed application developement. Hyperdatabases provide for higher order data independence in analogy to databases twenty years ago: Application programming will be at a level where availability, correctness, well-defined termination, caching, materialization, and replication of complete software components is merely an issue of the hyperdatabase system, taken off from the responsibility of the programmer. Many issues must be tackled in order to realize the vision. In the second part we have presented some concrete projects at ETHZ in order to exemplify some of the issues. We have summarized transactional process management as an evolution from database transactions. We have shortly described PowerDB as an HDB project with the main objective of demonstrating that coordinating many database systems in a cluster of workstations pays off with respect to scalability and that there is no need for sacrificing correctness. The coordination of heterogeneous, specialized components was the main issue of the image system project. The VA-File and feature extraction are examples of very special components that nicely can be coordinated by an HDB.

References

[Alo+97] Alonso, G., Blott, S., Fessler, A., Schek, H.-J.: *Correctness and Parallelism in Composite Systems.* In: Proc. of the 16th Symp. on Principles of Database Systems (PODS'97), Tucson, Arizona, May 12-15, 1997

[AHST97] Alonso, G., Hagen, C., Schek, H.-J., Tresch, M.: *Distributed Processing over Stand-alone Systems and Applications*, In: Proc. of 23rd International Conference on Very Large Data Bases (VLDB'97), August, 1997, Athens, Greece

[Alo+99a] Alonso, G., Fessler, A., Pardon, G., Schek, H.-J.: *Transactions in Stack, Fork, and Join Composite Systems.* In: Proc. of the 7th Int. Conf. on Database Theory (ICDT'99), Jerusalem, Israel, Jan. 10-12, C. Beeri, P. Buneman (Eds.), LNCS, Vol. 1540, Springer-Verlag, 1999, pp. 150-168.

[Alo+99b] Alonso, G., Fessler, A., Pardon, G., Schek, H.-J.: *Correctness in General Configurations of Transactional Components.* In: Proc. of the 18th ACM SIGMOD-SIGACT-SIGART Symp. on Principles of Database Systems (PODS'99), Philadelphia, Pennsylvania, May 31-June 2, ACM Press, New York, 1999, pp. 285-293.

[BGRS00] Böhm, K., Grabs, T., Röhm, U., Schek, H.-J.: *Evaluating the Coordination Overhead of Synchronous Replica Maintenance in a Cluster of Databases.* To appear in: Proc. of the 5th European Conference on Parallel Computing (Euro-Par 2000), Munich, Germany, August 2000.

[BMV96] Barbará, D., Mehrotra, S., Vallabhaneni, P.: *The Gold Text Indexing Engine,* In: Proc. of the Twelfth International Conference on Data Engineering (ICDE'96), February 26 - March 1, 1996, New Orleans, Louisiana, USA, pp. 172-179.

[Boe+98] Böhm, K., Ma, D., Nerjes, G., Schek, H.-J., Weber, R.: *Metadata Management with the HERMES Coordination Middleware*, ESPRIT project HERMES (no. 9141), Aug. 1998, Available at http://www-dbs.ethz.ch/~weber/paper/HERMESmeta.ps

[Boe00] Böhm, K.: *On Extending the XML Engine with Query Processing Capabilities.* In: IEEE Advances in Digital Libraries, 2000, Bethesda, Maryland, USA.

[BoK99] Boucher, K., Katz, F.: *The Essential Guide to Object Monitors,* John Wiley & Sons, New York etc., 1999.

[Bro99] Brodie, M.L.: *Que Sera, Sera: The Coincidental Confluence of Economics, Business, and Collaborative Computing,* Proc. of the 15th International Conference on Data Engineering, Sydney, Austrialia, March 1999, pp. 2-3

[CD96] Chen, Q., Dayal, U.: *A Transactional Nested Process Management System*. In: Proc. of the 12th International Conference on Data Engineering (ICDE'96), New Orleans, Louisiana, February 1996, pp. 566-573.

[CLL85] Carey, M.J., Livny, M., Lu, H.: *Dynamic Task Allocation in a Distributed Database System.* In: Proc. of the 5th IEEE Int. Conf. on Distributed Computing Systems (ICDCS), Denver, Colorado, May 1985.

[CORBA] http://www.corba.org

[DFS99] Deutsch, A., Fernandez, M., Suciu, D.: *Storing Semistructured Datawith STORED,* In: Proc. of the ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadephia, Pennsylvania, USA, pp. 431-442.

[FG+97] Fox, A. Gribble, S.D., Chawathe, Y., Brewer, E.A., Gaulthier, P.: *Cluster-Based Scalable Network Services.* In: Proc. of the 16th ACM Symposium on Operating System Principles (SOSP'97), St. Malo, France, 1997, pp. 78 - 91.

[FlK99] Florescu, D., Kossmann, D.: *Storing and Querying XML Data using an RDBMS.* In: IEEE Data Engineering Bulletin 1999, 22(3), pp. 27-34.

[GBS99] Grabs, T., Böhm, K., Schek, H.-J.: *A Document Engine on a DB Cluster.* In: Proc. of the 8th Int. Workshop on High Performance Transaction Systems (HPTS'99), Asilomar, California, Sept. 26-29, 1999.

[GBS00] Grabs, T., Böhm, K., Schek, H.-J.: *A Parallel Document Engine Built on Top of a Cluster of Databases -- Design, Implementation, and Experiences* --. Techn. Report No. 340, Dept. of Computer Science, ETH Zurich, April 2000.

[GHOS96] Gray, J., Helland, P., O'Neill, P.E., Shasha, D.: *The Dangers of Replication and a Solution*. In: Proc. of the SIGMOD Conference, pp. 173-182, 1996.

[ink96] Inktomi Corp., *The Inktomi Technology behind HotBot*, http://www.inktomi.com/products/-network/traffic/tech/clustered.html, 1996

[KaR96] Kamath, M., Ramamritham, K.: *Efficient Transaction Support for Dynamic Information Retrieval Systems.* In: Proc. of the 19th annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'96), Zurich, Switzerland, 1996.

[KaS96] Kaufmann, H., Schek, H.-J.: *Extending TP-Monitors for Intra-Transaction Parallelism.* In: Proc. of the 4th Int. Conf. on Parallel and Distributed Information Systems (PDIS'96), Miami Beach, Florida, USA, Dec. 18-20, 1996, p. 250-261

[MRSK92] Mehrotra, S., Rastogi, R., Silberschatz, A., Korth. H.: *A Transaction Model for Multidatabase Systems*. In: Proc. of the 12th International Conference on Distributed Computing Systems (ICDCS'92), Yokohama, Japan, June 1992, pages 56-63,

[NSSW94] Norrie, M., Schaad, W., Schek, H.-J., Wunderli, M.: *CIM Through Database Coordination*. In: Proc. of the 4th International Conference on Data and Knowledge Systems for Manufacturing and Engineering (DKSME'94), Hong Kong, May 1994, pp. 668-673.

[RB99] Röhm, U., Böhm, K: *Working Together in Harmony – An Implementation of the CORBA Object Query Service and its Evaluation.* In: Proc. of the 15th Int. Conf. on Data Engineering (ICDE 1999), Sydney, Australia, March 1999.

[RBS00] Röhm, U., Böhm, K., Schek, H.-J.: *OLAP Query Routing and Physical Design in a Database Cluster.* In: Proc. of the 7th Conf. on Extending Database Technology (EDBT 2000), Konstanz, Germany, March 27-31, 2000.

[Regr] Seminar talk "A sketch of Regres "http://www.cs.berkeley.edu/~gribble/-summaries/talks_seminars/regres.html

[RNS96] Rys, M., Norrie, M.C., Schek, H.-J.: Intra-Transaction *Parallelism in the Mapping of an Object Model to a Relational Multi-Processor System.* In: Proc. of the 22nd VLDB Conf., Mumbai (Bombay), India, Sept. 3-6, 1996, p. 460-471

[Roc71] Rocchio Jr., J.J.: *Relevance Feedback in Information Retrieval, The SMART Retrieval System: Experiments in Automatic Document Processing*, Prentice Hall, 1971, Englewood Cliffs, New Jersey, USA, pp. 313--323.

[RHOM98] Rui, Y., Huang, T.S., Ortega, M., Mehrotra, S.: *Relevance Feedback: A Power Tool in Interactive Content-Based Image Retrieval,* In: IEEE Transactions on Circuits and Systems for Video Technology, Special Issue on Segmentation, Description and Retrieval of Video Content, 8(5), 1998, pp. 644-655.

[ScW00] Schek, H.-J., Weber, R.: *Higher Order Databases and Multimedia Information*, In: Proc. of the Swiss/Japan Seminar Advances in Database and Multimedia, Kyoto, Japan, Febr. 2000.

[SAS99] Schuldt, H., Alonso, G., Schek, H.-J.: *Concurrency Control and Recovery in Transactional Process Management.* In: Proc. of the 18th ACM SIGMOD-SIGACT-SIGART Symp. on Principles of Database Systems (PODS'99), Philadelphia, Pennsylvania, May 31-June 2, ACM Press, New York, 1999, pp. 316-326.

[SPS00] Schuldt, H., Popovici, A., Schek, H.-J.: *Automatic Generation of Reliable E-Commerce Payment Processes*. In: Proc. of the 1st International Conference on Web Information Systems Engineering (WISE'00), Hong Kong, China, June 2000.

[SSA99] Schuldt, H., Schek, H.-J., Alonso, G.: *Transactional Coordination Agents for Composite Systems*. In: Proc. of the International Database Engineering and Applications Symposium (IDEAS'99), Montréal, Canada, August, 1999, pp. 321 - 331.

[Si+96] Silberschatz, A. et al.: *Strategic directions in database systems – breaking out of the box.* ACM Computing Surveys, Vol. 28, No. 4, Dec. 1996, pp. 764-778.

[SSW90] Schek, H.-J., Scholl, M.H., Weikum, G.: *From the KERNEL to the COSMOS: The Database*

Research Group at ETH Zurich. Techn. Report No. 136, Dept. of Computer Science, ETH Zurich, July 1990

[SSW95] Schaad, W., Schek, H.-J., Weikum, G.: *Implementation and Performance of Multi-level Transaction Management in a Multidatabase Environment.* In: 5th Int. Workshop on Research Issues on Data Engineering: Distributed Object Management, RIDE-DOM'95, Taipei, Taiwan, March 1995, p. 108-115

[ST+99] Shanmugasundaram, J, Tufte, K., He, G., Zhang, C., DeWitt, D., Naughton, J.: *Relational Databases for Querying XML Documents: Limitations and Opportunities.* In: Proc. of the 25th Int. Conf. on Very Large Data Bases (VLDB'99), Sept. 7-10, 1999, Edinburgh, Scotland, UK, pp. 302-314.

[SWS91] Schek, H.-J., Weikum, G., Schaad, W.: *A Multi-Level Transaction Approach to Federated DBMS Transaction Management.* In: Proc. of the First Int. Workshop on Interoperability in Multidatabase Systems, (IMS'91), Kyoto, April 1991

[SWY93] Schek, H.-J., Weikum, G., Ye, H.: *Towards a Unified Theory of Concurrency Control and Recovery.* In: 12th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS), Washington, DC, May 1993 (Appeared as Techn. Report No. 190, Dept. of Computer Science, ETH Zurich, Dec. 1992)

[VD98] Vogels, W., Dumitriu, D., et al*.: The Design and Architecture of the Microsoft Cluster Service - A Practical Approach to High-Availability and Scalability.* FTCS 1998.

[VLDB] http://www.vldb.org/

[W3C98] The World Wide Web Consortium: *Extensible Markup Language (XML) 1.0 - W3C Recommendation 10-February-1998. Available at: http://www.w3.org/TR/1998/REC-xml-19980210*

[Web+99] Weber, R., Bolliger, J., Gross, T., Schek, H.-J.: *Architecture of a Networked Image Search and Retrieval System.* In: Proc. of the 8th Int. Conf. on Information Knowledge Management (CIKM'99), Kansas City, Missouri, Nov. 2-6, ACM Press, New York, 1999, pp. 430-441.

[WeB00] Weber, R., Böhm, K.: *Trading Quality for Time with Nearest-Neighbor Search,* Proc. of the 7th Conf. on Extending Database Technology (EDBT 2000), Konstanz, Germany, pp. 21-35.

[Wei91] Weikum, G.: *Principles and Realization Strategies of Multi-Level Transaction Management.* In: ACM Transactions on Database System, Vol. 16, No. 1, March 1991, pp. 132-180

[WeS92] Weikum, G., Schek, H.-J.: *Concepts and Applications of Multilevel Transactions and Open Nested Transactions.* In: A.K. Elmagarmid (Ed.), Database Transaction Models for Advanced Applications, Morgan Kaufmann, San Mateo, CA, 1992

[WeS99] Weber, R., Schek, H.-J.: *A Distributed Image-Database Architecture for Efficient Insertion and Retrieval.* In: Proc. of the 5th Int. Workshop on Multimedia Information Systems (MIS'99), Indian Wells, California, Oct. 21-23, L. Golubchik, V. J. Tsotras (Eds.), pp. 48-55.

[WR92] Wächter, H., Reuter, A.: *The ConTract Model*, chapter 7, pp. 219-263. In: A.K. Elmagarmid (Ed.), Database Transaction Models for Advanced Applications, Morgan Kaufmann, San Mateo, CA, 1992

[WSB98] Weber, R., Schek, H.-J., Blott, S.: *A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces.* In: Proc. of the 24th Int. Conf. on Very Large Data Bases (VLDB'98), New York, USA, Aug.24-27, 1998.

[YCDI87] Yu, P.S., et. al.: *Analysis of Affinity Based Routing in Multi-System Data Sharing.* Performance Evaluation, 7:87-109, 1987.

[ZNBB94] Zhang, A., Nodine, M., Bhargava, A., Bukhres. O.: *Ensuring Relaxed Atomicity for Flexible Transactions in Multidatabase Systems*. In: Proc. of the ACM SIGMOD International Conference on Management of Data (SIGMOD'94), Minneapolis, Minnesota, May 1994, pp. 67-78.