

Challenges and Trends in Information Management

From a customer perspective, three main dimensions are relevant when evaluating and procuring database systems: functionality, performance, and total cost of ownership. Traditionally, database research has focused a lot on performance improvements of database systems, but less on new functionality and reducing the total cost of ownership. In this paper, we give our perspective on these three dimensions based on our experience in an industrial research laboratory. The paper is not intended to give a comprehensive overview of all activities, nor is it intended to provide an in-depth discussion of the research work we illustrate. Instead it highlights a set of activities at IBM's Almaden Research Center and outlines open research challenges that could be tackled by universities in Germany and elsewhere.

With respect to the performance dimension, systems are being designed to be more scalable, by utilizing hardware support to evaluate queries close to the storage subsystem (Netezza), and by massively parallel systems like Google's map/reduce, which go beyond classical shared-nothing or shared-disk parallelism. That said performance seems to play a less important role in modern database systems as users are willing to trade performance for a reduction of total cost of ownership and – to a lesser extent – for an increase in functionality.

The functionality dimension of database systems has evolved from simple SQL queries to recursive queries, to data mining, and to support for semi-structured data (for example, XML). However, several areas remain insufficiently supported.

We see a future trend to enable support of unstructured information, and for combined storage and querying of semi-structured and unstructured data. In Section 1 we introduce AVATAR, a project aimed at discovering and extracting structured information buried in volumes of unstructured text such as emails and blogs. This information can then be exploited to drive the next generation of

search and business intelligence applications.

Another emerging trend is the demand for legislative compliance technology. In Section 2 we discuss Hippocratic Database technology as IBM's approach to tackle the enforcement of privacy regulations or confidentiality policies. Increasing legal regulations regarding privacy of personal data require support for compliance without impeding the flow of information. Whenever business-critical data is involved, companies want to protect the confidentiality of their data and share only on a need-to-know basis according to carefully crafted sharing policies.

Total cost of ownership has become an active research area. As computer systems become cheaper, manpower to maintain those systems is becoming the largest cost factor. In Section 3 we discuss some of the work carried out at IBM in order to make database systems more self-tuning. As a future manifestation of that trend we see that DBMSs will be offered as appliances, with little or no configuration required.

1 Semantic Search and High Precision Annotations

1.1 Problem Description

AVATAR Semantic Search tackles the problem of precision-oriented retrieval. A keyword query is interpreted as a set of precise queries in the context of information extracted from text. Consider the scenario of searching email. Suppose a user submits the keyword query [jeff phone]. A standard keyword search engine will interpret this query as »retrieve emails that contain the words jeff and phone.« However, such an interpretation will not return emails which mention a phone number but not the keyword »phone.« The goal of the AVATAR project is two fold: (i) to enable the discovery and extraction of structured information buried in volumes of unstructured text (such as emails, web pages, and blogs), and (ii) to

exploit this information to drive the next generation of search and business intelligence applications. In the following sections we present our work on both aspects.

1.2 The AVATAR Information Extraction System

Text analytics is a mature area of research concerned with the problem of automatically analyzing text to extract structured information. Examples of common text analytic tasks include *entity identification* (e.g., identifying persons, locations, organizations, etc.) [Cunningham 1997], *relationship detection* (e.g., person X works in company Y) [Nanda 2004] and *co-reference resolution* (identifying different variants of the same entity either in the same document or different documents) [McCarthy & Lehnert 1995]. Text analytics programs used for information extraction are called **annotators** and the objects extracted by them are called **annotations**. Traditionally, such annotations have been directly absorbed into applications. Moreover, a very important pre-requisite for the use of annotations in enterprise applications is *high precision*. Increasingly, due to the complex needs of today's enterprise applications (such as Community Information Management [Doan et al. 2006]), there is a need for infrastructure that enables information extraction, manages the extracted objects. Furthermore, to enable collaborative efforts to build high precision annotations, the infrastructure should allow sharing annotators among users and reusing existing annotators in different text corpora.

At the IBM Almaden Research Center we are currently building the AVATAR Information Extraction System (IES) to tackle the challenge of creating annotators and annotations. Every annotation object produced by an annotator is characterized by a well-defined set of operations. We refer to each such operation as a *rule*. A set of rules within an annotator that together produce an annotation object is called a *meta-rule*. As an example, consider a simple base annotator that identifies occurrences of person names in the text of a document. An example of a meta-rule used by such an annotator would be (informally) *MRI: look for the presence of a salutation such as Dr. or Mr. followed by a capitalized word*. Meta-rule

MRI would identify »Dr. Dasovich« as a candidate *Person* annotation.

Annotator Data Model

Each annotation produced by the AVATAR IES can be viewed as a structured object with a well-defined type. The overall output of an IES, called an annotation store, is a collection of such objects. Please refer to [Jayram et al. 2006] for a formal definition of an annotation store. Figure 1 shows a simple annotation store with one document object of type *Email* and three annotation objects. Each oval in the figure represents one object and is labeled as *A:B* where *A* is the ID of the object and *B* is the type. The rectangular boxes represent atomic attribute values. In this example the span type *S* contains a pair of integers *begin* and *end* that store the character offsets of the piece of the text corresponding to the annotation. For the purposes of this paper, we assume that every annotation is extracted from a single document and that the span is interpreted relative to the *text* of this document.

Annotators in AVATAR IES are classified into two categories based on their input:

Base Annotators operate over the document text, independent of any other annotator. E.g., base annotator *SimplePerson* identifies persons using two dictionaries – *Du* (containing unambiguous first or last names such as »jeff«, »dasovich«, etc.) and *Da* (containing ambiguous first or last names such as »bill«, »gray«). Following the template described in [Jayram et al. 2006] the base annotator *SimplePerson* works as follows:

1. The input document is tokenized and each individual token is added to the

set of features. Further, for each feature, an attribute *dictU* (resp. *dictA*) is set to true if the corresponding token matches an entry in the dictionaries *Du* or *Da*.

2. Apply a set of rules for identifying person names (i) a pair of features that are adjacent to each other in the document text and both of which are labeled with *Du* (e.g., jeff dasovich), (ii) a pair of features that are adjacent to each other in the document text and are labeled with *Du* and *Da* respectively (e.g., james gray), (iii) a feature that is labeled with *Du* (e.g., jeff), and (iv) a feature that is labeled with *Da* (e.g., gray).
3. Apply a consolidation rule *rd* that executes the following logic: If two candidates *o1* and *o2* are such that the *o1.span* contains *o2.span*, discard *o2*. This rule is applied repeatedly to produce the result of the annotator.

[Jayram et al. 2006] describes the functionality of the annotator in more details.

Derived Annotators operate on document text as well as the annotation objects produced by other annotators. E.g., the derived annotator *PersonPhone* identifies people’s phone numbers from documents. This annotator takes in as input the document and a set of *Person*, *Phone* and *ContactPattern* annotations identified by Base annotators. Here (i) *Person* annotations are produced by a person annotator such as *SimplePerson*, (ii) *Phone* annotations are produced by an annotator that identifies telephone numbers, and (iii) *ContactPattern* annotations are produced by an annotator that looks for occurrences of phrases such as »at«, »can be (reache | called) at« and »’s number is«. Given the

se inputs, the *PersonPhone* annotator is fully described by the following two rules:

- If a triple (*Person*, *ContactPattern*, *Phone*) appear sequentially in the document, create a candidate *PersonPhone* annotation. An example instance identified using this rule is shown in Figure 1.
- Consolidation rule *rd* as described earlier.

The current implementation of AVATAR IES uses the UIMA [Ferrucci & Lally 2004] workflow engine to execute annotators. Annotations produced in AVATAR IES are stored in an annotation store implemented using a relational database (DB2). The store allows multiple derived annotators to be executed without having to re-execute the base annotators.

1.3 The AVATAR Semantic Search System

Our primary focus will be the task of bridging the gap between the keyword queries that users submit and the structured queries that model their intent. *AVATAR Semantic Search* uses the notion of *keyword query interpretation* – a process by which end-user keyword queries are automatically converted into one or more precise queries called *interpretations*. Each interpretation, when executed over the annotation store, produces a set of documents as result.

Runtime architecture: Figure 2a shows the runtime architecture for AVATAR Semantic Search consisting of a Semantic Optimizer and a User Interaction Engine. The semantic optimizer takes a keyword query as input and produces a ranked list of semantically meaningful interpretations of that query. The top ranked interpretations produced by the semantic optimizer are handed over to the *user interaction engine*, which is a suite of tools for presenting and displaying interpretations and result documents to the user. These two components use a pair of indexes as shown in the figure 2b (i) a *translation index* to help identify the set of relevant interpretations for a given keyword query, and (ii) a *data index* over which individual interpretations are executed.

Translation index: It is used by the semantic optimizer to retrieve the possible »meanings« of an individual key-

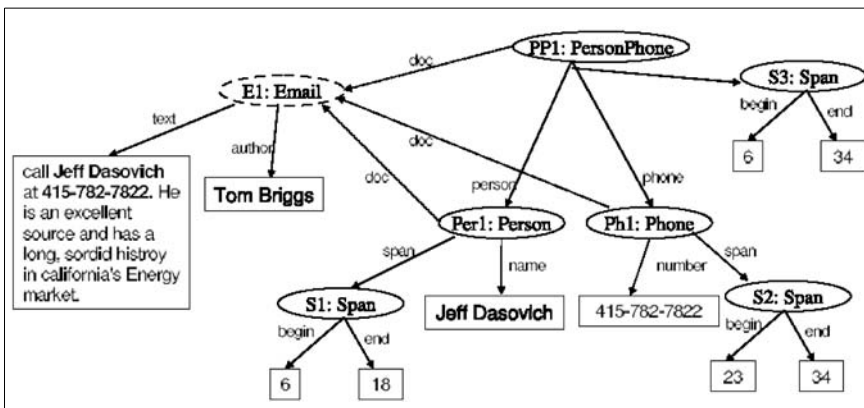


Fig. 1: Annotation Store

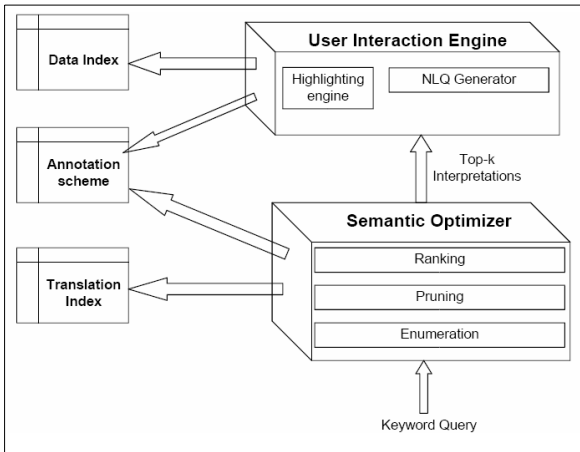


Fig. 2a: Runtime architecture

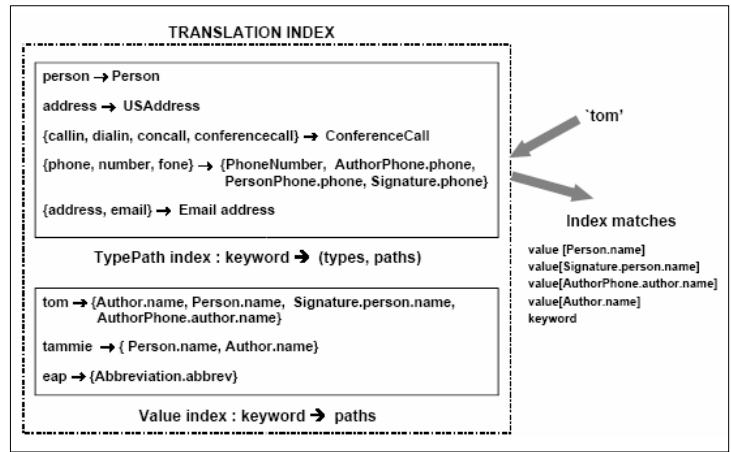


Fig. 2b: Translation index

word. Given a keyword, the index returns the (i) concepts in the schema, such as types and paths, that match the keyword (based on keywords provided by domain experts for types and paths in the schema), and (ii) paths where the keyword appears as a data value in the data set.

Data index: It is used to execute the interpretations generated by the semantic optimizer. We implemented the data index using the IBM OmniFind [WebSphere 2006] indexing engine that not only supports XPath queries but also incorporates all of the standard text retrieval features such as stemming, multi-word tokenization, stop word dictionaries, etc.

Semantic optimizer: The key component of the *AVATAR Semantic Search* runtime architecture is the *semantic optimizer* (fig. 2a). Internally, the semantic optimizer is organized as a sequence of the following three stages:

Enumeration: The concept of justification is used to formally define the class of queries over the annotation store that represents valid interpretations of a keyword query. In the enumeration step, user keywords are matched against the translation index as shown in figure 2b. The resulting matches are used to generate every possible *justified* interpretation of a keyword query.

Pruning: Even for annotation stores with a small number of types, the set of possible justified interpretations can be extremely large. The pruning step uses domain specific constraints over the annotation schema to eliminate several of these interpretations,

see [Kandogan et al. 2006] for example constraints.

Ranking: After pruning, every remaining interpretation is potentially of interest to a user who submitted the original keyword query. The goal of ranking is to generate an ordering of these interpretations so that the more likely interpretations are ranked higher, see [Kandogan et al. 2006] for an example on ranking.

1.4 Challenges

AVATAR IES scales to a large number of high precision annotators. We are currently building a framework to map such annotators to queries in a data base system. Deriving precision estimates in such a database system presents a significant challenge for probabilistic database systems. This is the focus of our ongoing work in this area, such as in [Kandogan et al. 2006, Burdick et al. 2005].

2 Enforcing Privacy and Confidentiality of Data

2.1 Problem Description

Several governments have passed data protection laws to govern the processing and movement of personal data. For example, the European Union passed the 1995 European Union Directive on Data Protection [EU Directive 1995] requiring its member states to adopt laws to protect personally identifiable information. For obvious reasons privacy of health care information has received significant legislative attention in countries such as the

US, Canada, Australia, and Japan [Agrawal & Johnson 2006]. As a consequence, health care providers in these countries need support to comply with regulations on which data (addresses, medical history, drug prescriptions) can be shared with whom (physicians, pharmacists, researchers) for which purpose (treatment, research, marketing).

To stay competitive in a global market, companies are increasingly forced to share information among each other. For example, tracking individual RFID-tagged items throughout supply chains of independent enterprises is an emerging trend in many industries. To completely track an item data has to be shared across multiple, possibly competing, enterprises. The need to simultaneously compete and cooperate requires new technologies that allow companies to share their traceability data while maintaining complete sovereignty over what data is shared and with whom [Agrawal et al. 2006].

The goal of IBM's data privacy research is to build next-generation information systems that preserve privacy and confidentiality of data without impeding the flow of information.

2.2 The Hippocratic Database

Traditionally, database systems have been designed to guarantee efficient storage and access of information. Most current commercial database systems still use the authorization model System R introduced in 1976 [Astrahan et al. 1976]. Consequently, privacy and confidentiality of data is mostly handled outside the database system in individual applica-

tions. This approach has two severe drawbacks: First, functionality needed by multiple applications is re-implemented in each application. Second, the control over data is moved away from the database, thus increasing the potential for data abuse and policy violation.

Hippocratic database (HDB) [Agrawal et al. 2002] is a vision that embraces the responsibility for the privacy of data managed as a fundamental tenet of the database, realized using a suite of technologies to manage disclosure of data. We describe four such technologies: active enforcement, compliance auditing, sovereign information sharing, and privacy-preserving data mining.

Active Enforcement: Active enforcement [Lefevre et al. 2004] ensures that applications accessing data within a Hippocratic database will adhere to data-disclosure policies. Active enforcement takes place in three phases: policy installation, user preference collection, and data retrieval (see fig. 3). In the policy installation phase an organization specifies its data-sharing practices using a policy specification language such as P3P [Cranor et al. 2002]. Policies may be specified with specific query-purpose and result-recipient in mind. The policy specification is shredded and stored inside the database as relations. In the next phase, individual user preferences such as opt-in and opt-out choices are collected. The user preferences are expressed in XPref [Agrawal et al. 2003b], an XPath-based language. In the data retrieval phase, application queries are modified based on the policy specification and user preferences. These modifications are done at the SQL interface level (for example, using a special JDBC driver). The modified query is able to restrict access at individual cell level without requiring any modification of applications.

Compliance Auditing: Compliance auditing (CA) [Agrawal et al. 2004] gives companies the ability to retroactively demonstrate that their data-management practice did comply with the declared policies. A naive implementation logs, for each query, all records accessed by the query. Our approach logs only query strings and the rest of the query-processing is unencumbered (see fig. 4). Offline, the database recovery log is used to create backlog tables, which allow reconstruction of any past state of the database. To

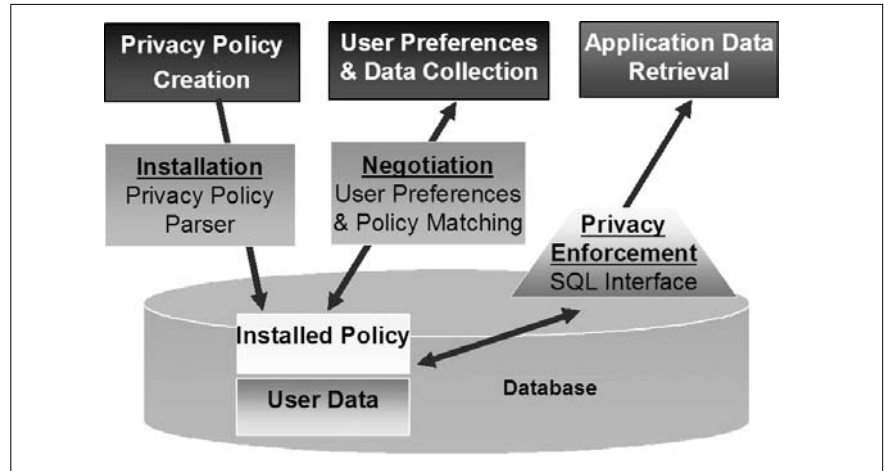


Fig. 3: HDB Active Enforcement

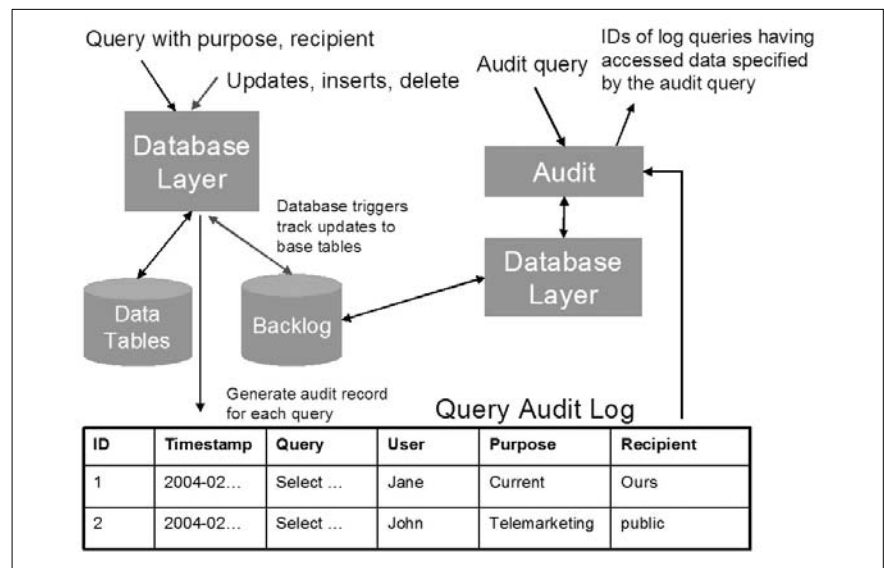


Fig. 4: HDB Compliance Auditing

perform an audit, the auditor formulates an expression specifying the data whose disclosure is to be tracked. Audit expressions are similar to SQL queries, and allow audits to be performed at individual cell level. When an audit expression is received, it is statically analysis to select candidate queries that could potentially have disclosed the data being audited. The audit expression is then combined with the candidate queries, and run against the backlog database. The final result of auditing returns the precise set of queries that accessed the specified data.

Sovereign Information Sharing: In certain situations, enterprises are interested in sharing query results, but they do not want to reveal all records accessed to compute the result (see fig. 5 for an illus-

tration). For example, airlines and security agencies want to know whether a suspect is on board a plane. However, neither does the security agency want to reveal its list of suspects, nor the airline its list of passengers. Sovereign Information Sharing (SIS) [Agrawal et al. 2003a] provides a new paradigm to meet the requirements for such information sharing across autonomous entities. In this model, the execution of a query over two or more databases reveals *only the result* of the query to all entities involved. SIS applies a set of commutative encryption functions to the data at each location. The multiply encrypted values are then compared, and the query results provided, without compromising the privacy of any data set. SIS can be seamlessly integrated into existing environments without any anonymization

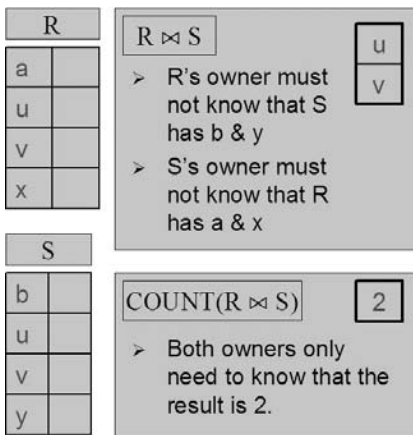


Fig. 5: HDB Sovereign Information Sharing

of data or without the need for a trusted third-party.

Privacy-Preserving Data Mining:

In some sense privacy and data mining have conflicting goals: Privacy aims at protecting information whereas data mining aims at discovering information. Privacy-Preserving Data Mining [Agrawal & Srikant 2000] resolves this dilemma by exploiting the difference between the level at which people care about privacy, i.e. individual data, and the level at which data mining algorithms operate, i.e. aggregated data. User data is randomized such that it cannot be recovered at the individual level with reasonable precision, but the original data distribution can still be reconstructed at the aggregate level. Thus, classification and association-rule mining can be performed on top of privacy-preserved data with only a small loss in accuracy [Evfimievski 2002].

2.3 Research Challenges

Policy specification: HDB relies on the fact that privacy and confidentiality constraints can be captured in the form of a policy, using a policy specification language. The policy specifications must be complete, correct, concise, and clear to three participants: The *policy maker* requires the policy to be easy to compose and to precisely capture her intent. The *human policy readers* want it to be easier to read and understand than long legal texts. *HDB* requires the policy to be in a format that can be efficiently interpreted. Designing a policy language that satisfies all three participants is a major challenge.

Currently, policy languages do not capture privacy constraints that relate to multiple pieces of information, e.g. an en-

terprise is willing to share information I or H separately, but it is not willing to reveal I and H together. When such constraints can be expressed, policy enforcement and auditing techniques need to be extended to detect inferences possible by issuing multiple (correlating) queries.

Sticky policies: HDB allows company X to enforce its confidentiality policy when company Y requests information I . However, it does not give X control over what happens to I once it is released to Y . If X 's policy states that it is willing to share information I with Y but with no other company, this policy should also be enforced by Y . One way is to stick a piece of policy to I telling Y not to share I with anyone else. Some of the challenges that would have to be addressed to enable such a solution are: a standardized format for sticky policies, a way to generate sticky policies given a query result, and a way to import sticky policies into another database.

Data Lifecycle Management: Privacy protection also needs to consider data lifecycle issues such as retention period. Some regulations require organizations to retain some kinds of data for a minimum amount time. Similarly, there are regulations and policies that require organizations to delete some kinds of data after a certain period of time. For example, the Health Insurance Portability and Accountability Act [HIPAA 1996] requires hospitals providing certain services to keep patient records for a period of five years. In contrast, patient data collected at a hospital for a specific research purpose might have to be deleted when the research ends. In some cases, the same data might be collected for purposes with conflicting retention requirements. Consequently, managing the lifecycle of such data, especially those that additionally require privacy enforcement and compliance auditing, becomes harder.

3 Autonomic DBMS

3.1 Requirements for an Autonomic DBMS

To reduce the total cost of ownership database systems should be more self-tuning. We divided the operational requirements of autonomic DBMS in the four categories self-configuring, self-healing, self-optimizing, and self-protecting.

Self-configuring refers to function that connects and configures components so that technology is operational with minimal human intervention. For database servers, this can include server and storage topology configuration, installation, initial database design, configuration of the database server (there are several dozen possible configuration options), and physical database design. Self-configuring also includes the ability for a database server to be easily deployed, configured, and interconnected with other middleware and applications.

Self-healing refers to the ability of a system to detect problems or problem trends (which have not yet manifested as problems) and either take corrective action or provide advice and automated notification to an end user, who can then take corrective action. Even with all the sophistication and redundancy in database systems, things can and will occasionally fail, which makes the automated problem determination aspect of self-healing a critical domain as well. In automated problem determination, an automatic system detects problem trends (such as disk storage filling up at a detectable rate, from which a storage constraint can be projected) or detects and notifies the DBA when sudden errors arise.

Self-optimizing is the domain of system performance tuning. For databases, this includes compiling queries so that they perform well when run, tuning utilities so that they maximize the available resources, and throttling background processes to avoid undesirable impact to the production workload. Self-optimizing also includes the daily maintenance of the database, which, if neglected, will generally lead to system performance degradation or storage waste. This includes, for example, regular backups, collection of data statistics, and data defragmentation and reclustering.

The final functional category is **Self-protecting**, which describes a system's ability to anticipate, identify, and protect against security attacks. The scope varies from static defenses such as virus protection and data encryption to more advanced dynamic defenses such as behavior analysis, where the actions taken by the users accessing the RDBMS are analyzed for negative behaviors, tracked, and adaptively unwound.

3.2 A Self-Optimizing DBMS

A database management system has various ways of processing any given SQL query. For instance, it could use a hash-join or a nested-loops join to combine two tables. It also has to select one or more of many available indexes to evaluate a selection predicate and to determine the order in which tables should be joined together. It is the task of the query optimizer of a relational DBMS to choose the optimal query execution plan (QEP) for any given query. In order to achieve that, the query optimizer relies on a mathematical model of the cost of each query execution plan, which in turn relies on proper estimates of the sizes of intermediate steps (cardinalities) in each plan that is considered. This cardinality model is based on statistics about the distribution of values in the database.

Tuning query plans, and specifically keeping the necessary statistics for the query optimizer available and up to date, is a major area of concern in day-to-day database operations. At the IBM Almaden Research Center, we have carried out a vast body of research on autonomic computing for information management [Garcia-Arellano et al. 2006] and specifically on automatically collecting, maintaining, and optimally using statistics in the query optimizer [Markl et al. 2003].

In order to automatically maintain statistics for the query optimizer, we explored closed loop tuning in our LEO (Learning Optimizer, see [Stillger et al. 2001]) and POP (Progressive Optimization, see [Markl et al. 2004]) projects. Both LEO and POP adjust the query optimizer's model based on actual information from running queries. POP uses that information to determine if a *currently running query* is suboptimal, and triggers re-optimization of the query in that case. LEO persists the learned information to remedy the errors in the optimizer's model for *future queries*. In contrast to ASP, both LEO and POP implement a fully closed loop, in that they do not merely give recommendations, but immediately alter the optimizer's model.

Both LEO and POP use results from actual executions of queries to validate the optimizer's model incrementally, deduce what part of the optimizer's model is in error, and compute adjustments to the optimizer's model for future query opti-

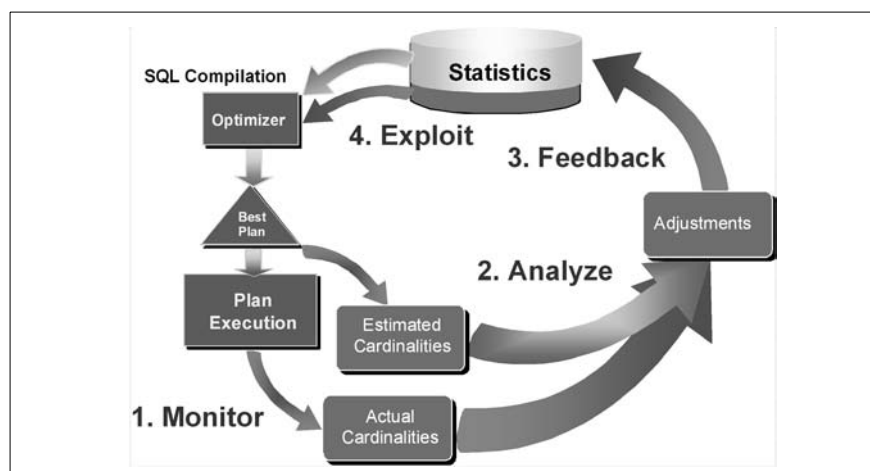


Fig. 6: Deferred Learning

mizations. Deferred learning with LEO works under the assumption that future queries are similar to previous queries, i.e., they share one or several predicates.

The LEO feedback loop is comprised of four steps: monitoring, analysis, feedback, and feedback exploitation. The monitoring component saves at query compilation time the cardinality estimates derived by the optimizer for the best (i.e., least-cost) plan, and during query execution saves the actual cardinalities observed for that plan. The analysis component uses the monitored information to identify modeling errors and compute corrective adjustments. This analysis is a standalone process that may be run separately from the database server, and even on another system. The feedback component modifies the catalog statistics of the database according to the learned information. The exploitation component closes the feedback loop by using the learned knowledge in the system catalog to provide adjustments to the query optimizer's cardinality estimates.

The four components can operate independently, but form a consecutive sequence that constitutes a continuous learning mechanism by incrementally capturing plans, monitoring their execution, analyzing the monitor output, and computing adjustments to be used for future query compilations. This mechanism enables deferred learning, since only future queries will benefit from the feedback.

The monitored cardinalities need not be used for subsequent queries alone. If the actual cardinalities are significantly different from the estimated cardinalities,

the chosen query plan could be highly sub-optimal. POP uses this knowledge immediately by dynamically re-optimizing the current query and changing its execution plan, if all of the rows for an intermediate result are materialized before proceeding at any point in the plan. In that way POP can improve queries under execution, while reusing the work already done.

POP combines a plan optimality criterion with checkpoints, runtime monitoring and a sophisticated matching of intermediate results. It can this way catch badly performing queries and improve them, while still imposing only a negligible overhead on well performing queries. During access plan selection, POP determines criteria for estimated parameters that are required to hold if the plan is to be the optimal one. The current prototype uses only the estimated cardinality, which is the most important parameter and also the one subject to the gravest estimation error. It computes the validity range around it, an interval that describes for which cardinality range the current plan is truly the optimal one. It then places CHECK operators at strategic points, which in turn validate during plan execution that the actual cardinality, obtained from the runtime monitor, is within the validity range. If this is not the case, all intermediate results from fully materialized points are retained and the optimizer is called again. The actual cardinalities from the aborted query execution are made available to the optimizer so that it is able to develop a better plan, which is not subject to the estimation error that caused the re-optimization. Note that this makes

POP suitable for any source of cardinality estimation error, be it bad statistics, wrong assumptions, or parameter markers. The retained intermediate results are treated as materialized views, also called materialized query tables (MQT) or automatic summary tables in DB2. The optimizer has the cost-based choice to match them back into the plan, enabling the query to basically continue from the point it was aborted for re-optimization, avoiding the re-execution of previously executed parts. A subset of our work has already made it into the IBM DB2 product [Aboulnaga et al. 2004], and we are currently working on actively extending the work to shared-nothing and federated environments [Ewen et al. 2005].

3.3 Challenges

While we have been successful with our research in autonomic query optimization, several challenges still remain: One big problem is how to maintain robustness in the light of an autonomic system. We have to ensure that a system converges to a stable point, and that performance becomes predictable. Customers prefer predictable performance over overall best performance. This may mean to alter the cost model of the query optimizer, to favor robust plans (plans that perform well even in the case of uncertainty of, for instance, cardinality estimates) over best plans. While some work has been carried out in that area, suggesting to consider estimates as expected values of a probability distribution, there has been no conclusive solution to that problem yet.

Other challenges arise when new functionality is added. Total cost of ownership has to be kept low, when databases support more complex data types, including semi-structured and unstructured data. This may mean to design robust runtime operators and thus remove the need of proper statistics at compile time of a query, effectively pushing optimization into the runtime system. The same holds true when query processing is extended to web-sources to enable mash-ups of information in the Web 2.0 style.

4 Summary

The next generation DBMS has to accomplish two almost competing goals. On the one hand the DBMS needs to em-

brace a multitude of new functionalities to support emerging data types and application needs. In this article, we have discussed two such future functionalities: support for semantic query processing and protection of data privacy. On the other hand, there is a need to reduce a DBMS' complexity, or hide its complexity from its users. We have discussed self-optimization as a way to reduce the maintenance cost of a database and gave an outlook how the total cost of ownership can be reduced.

We believe the three research areas illustrated in this article are of practical relevance, and there are opportunities for new discoveries and improvements in these areas. We hope this article encourages readers to learn more about these topics; perhaps to even conduct research in these areas.

References

- [Aboulnaga et al. 2004] *Aboulnaga, A.; Haas, P. J.; Lightstone, S.; Lohman, G. M.; Markl, V.; Popivanov, I.; Raman, V.*: Automated Statistics Collection in DB2 UDB. In: VLDB, 2004, pp. 1146-1157.
- [Agrawal & Johnson 2006] *Agrawal, R.; Johnson, C.*: Securing Electronic Health Records without Impeding the Flow of Information. In: International Medical Informatics Association Working Conference – Security in Health Information Systems, Dijon, France, April 2006.
- [Agrawal & Srikant 2000] *Agrawal, R.; Srikant, R.*: Privacy-Preserving Data Mining. In: SIGMOD, Dallas, Texas, USA, May 2000.
- [Agrawal et al. 2002] *Agrawal, R.; Kiernan, J.; Srikant, R.; Xu, Y.*: Hippocratic Databases. In: VLDB, Hong Kong, China, August 2002.
- [Agrawal et al. 2003a] *Agrawal, R.; Evfimievski, A.; Srikant, R.*: Information Sharing across Private Databases. In: SIGMOD, San Diego, California, June 2003.
- [Agrawal et al. 2003b] *Agrawal, R.; Kiernan, J.; Srikant, R.; Xu, Y.*: An XPath-based Preference Language for P3P. In: WWW, Budapest, Hungary, May 2003.
- [Agrawal et al. 2004] *Agrawal, R.; Bayardo, R.; Faloutsos, C.; Kiernan, J.; Rantzaou, R.; Srikant, R.*: Auditing Compliance with a Hippocratic Database. In: VLDB, Toronto, Canada, August 2004.
- [Agrawal et al. 2006] *Agrawal, R.; Cheung, A.; Kailing, K.; Schoenauer, S.*: Traceability across Sovereign, Distributed RFID Databases. To appear in: Proceedings of IDEAS 2006, New Delhi, India, December 2006.
- [Astrahan et al. 1976] *Astrahan, M. M.; Blasgen, M. W.; Chamberlin, D. D.; Eswaran, K. P.; Gray, J. N.; Griffiths, P. P.; King, W. F.; Lorie, R. A.; McJones, P. R.; Mehl, J. W.; Putzolu, G. R.; Traiger, I. L.; Wade, B. W.; Watson, V.*: System R: relational approach to database management. ACM Transactions on Database Systems (TODS), Volume 1, Issue 2, June 1976, pp. 97-137.
- [Burdick et al. 2005] *Burdick, D.; Deshpande, P.; Jayram, T. S.; Ramakrishnan, R.; Vaithyanathan, S.*: OLAP Over Uncertain and Imprecise Data. In: VLDB 2005.
- [Cranor et al. 2002] *Cranor, L.; Langheinrich, M.; Manchiori, M.; Presler-Marshall, M.; Reagle, J.*: Platform for Privacy Preferences 1.0 (P3P1.0) Specification, W3C Recommendation, April 2002.
- [Cunningham 1997] *Cunningham, H.*: Information extraction – a user guide. Technical Report CS-97-02, University of Sheffield, 1997.
- [Doan et al. 2006] *Doan, A.; Ramakrishnan, R.; Chen, F.; DeRose, P.; Lee, Y.; McCann, R.; Sayyadian, M.; Shen, W.*: Community information management. In: IEEE Data Engineering Bulletin, March 2006.
- [EU Directive 1995] European Union Directive on Data Protection. Official Journal of the European Communities, 23 November 1995, No L. 281, p. 31.
- [Evfimievski 2002] *Evfimievski, A.*: Randomization in Privacy-Preserving Data Mining. In: SIGKDD Explorations: Newsletter of the ACM Special Interest Group on Knowledge Discovery and Data Mining, 4(2), December 2002, pp. 43-48.
- [Ewen et al. 2005] *Ewen, S.; Ortega-Binderberger, M.; Markl, V.*: A Learning Optimizer for a Federated Database Management System. In: BTW, 2005, pp. 87-106.
- [Ferrucci & Lally 2004] *Ferrucci, D.; Lally, A.*: UIMA: An architectural approach to unstructured information processing in the corporate research environment. In: Natural Language Engineering, June 2004.
- [Garcia-Arellano et al. 2006] *Garcia-Arellano, C. M.; Lightstone, S.; Lohman, G. et al.*: Autonomic Features of the IBM DB2 Universal Database for Linux, Unix, and Windows. In: IEEE Transactions on Systems, Man, and Cybernetics 36(3), May 2006.
- [HIPAA 1996] Health Insurance Portability and Accountability Act of 1996. United States Public Law 104-191.
- [Ilyas et al. 2004] *Ilyas, I. F.; Markl, V.; Haas, P. J.; Brown, P.; Aboulnaga, A.*: Cords: Automatic Discovery of Correlations and Soft Functional Dependencies. In: SIGMOD, 2004, pp. 647-658.
- [Jayram et al. 2006] *Jayram, T. S.; Krishnamurthy, R.; Raghavan, S.; Vaithyanathan, S.; Zhu, H.*: Avatar information extraction system. IEEE Data Engineering Bulletin, May 2006.
- [Kandogan et al. 2006] *Kandogan, E.; Krishnamurthy, R.; Raghavan, S.; Vaithyanathan, S.; Zhu, H.*: Avatar semantic search: a database approach to information retrieval. ACM SIGMOD 2006.
- [Lefevre et al. 2004] *Lefevre, K.; Agrawal, R.; Ercegovac, V.; Ramakrishnan, R.; Xu, Y.; DeWitt, D.*: Limiting Disclosure in Hippocratic Databases. In: VLDB, Toronto, Canada, August 2004.
- [Markl et al. 2003] *Markl, V.; Lohman, G. M.; Raman, V.*: Leo: An Autonomic Query Optimi-

zer for DB2. IBM Systems Journal, 42(1):98-106, 2003.

[Markl et al. 2004] *Markl, V.; Raman, V.; Simmen, D. E.; Lohman, G. M.; Pirahesh, H.*: Robust Query Processing through Progressive Optimization. In: SIGMOD, 2004, pp. 659-670.

[McCarthy & Lehnert 1995] *McCarthy, J. F.; Lehnert, W. G.*: Using decision trees for coreference resolution. In: IJCAI, pp. 1050-1055, 1995.

[Nanda 2004] *Nanda, K.*: Combining lexical, syntactic and semantic features with maximum entropy models for extracting relations. In: ACL, 2004.

[Stillger et al. 2001] *Stillger, M.; Lohman, G. M.; Markl, V.; Kandil, M.*: Leo – DB2's Learning Optimizer. In: VLDB, 2001, pp. 19-28.

[Websphere 2006] Websphere Information Integrator OmniFind Edition, 2006; www.ibm.com/software/data.



Karin Kailing is a member of the Intelligent Information Systems group at the IBM Almaden Research Center. She received her PhD from the University of Munich 2004. Her research interests include querying and mining RFID data.



Alexander Löser is a Postdoc in the group of unstructured information management at the IBM Almaden Research Center. His current work focuses on high precision enterprise search engines. Learn more about him at <http://cis.cs.tu-berlin.de/~aloeser>.



Volker Markl has been working at IBM's Almaden Research Center in San Jose, USA since 2001, conducting research in query optimization, indexing, and self-managing databases.

Volker Markl is spearheading the LEO project, an effort on automatic computing with the goal to create a self-tuning optimizer for DB2 UDB. He also is the Almaden chair for the IBM Data Management Professional Interest Community (PIC). From 1997 to 2000, Volker Markl worked for the Bavarian Research Center for Knowledge-Based Systems (FORWISS). Volker Markl is a graduate of the Technische Universität München, where he earned a Masters degree in Computer Science in 1995. He completes his PhD in 1999. He also earned a degree in Business Administration from the University Hagen, Germany in 1995. Since 1996, Volker Markl has published more than 30 reviewed papers at prestigious scientific conferences and journals and filed more than 10 patents.

Dr. Karin Kailing
Dr. Alexander Löser
Dr. Volker Markl
IBM Almaden Research Center
650 Harry Road
San Jose
CA 95120, USA
{kkailin, aloeser, marklv}@us.ibm.com
www.ibm.com

Can Türker, Gunter Saake

Objektrelationale Datenbanken

Ein Lehrbuch

2005, 564 Seiten, Broschur
€ 39,00 (D) · ISBN 3-89864-190-2



Das Buch behandelt die Datenmodellkonzepte und Anfragesprachen objektrelationaler Datenbanken. Diese Datenbanktypen sind der De-facto-Standard, der in führenden Produkten wie Oracle 10g und DB2 realisiert ist.

Das Buch zeigt, wie sich die objektrelationalen Konzepte in aktuellen Sprachen und Systemen wiederfinden. Der Entwurf objektrelationaler Datenbanken wird ebenso diskutiert wie die Entwicklung entsprechender Datenbankanwendungen über die Java-SQL-Schnittstellen JDBC und SQLJ mit Schwerpunkt auf der Ausnutzung objektrelationaler Erweiterungen. Daneben werden Implementierungsaspekte wie Speicher- und Indexstrukturen erörtert.

Auf der Buch-Website befinden sich u.a. Übungen und Beispiele aus dem Buch, SQL-Skripts und eine Foliensammlung für Dozenten.

dpunkt.datenbanken



dpunkt.verlag

Ringstraße 19 B
D-69115 Heidelberg
fon: 0 62 21 / 14 83 40
fax: 0 62 21 / 14 83 99
e-mail: bestellung@dpunkt.de
www.dpunkt.de