The Architecture of the Cornell Knowledge Broker*

Alan Demers, Johannes Gehrke, and Mirek Riedewald

Department of Computer Science, Cornell University
{ademers,johannes,mirek}@cs.cornell.edu

Abstract. Intelligence applications have to process massive amounts of data in order to extract relevant information. This includes archived historical data as well as continuously arriving new data. We propose a novel architecture that addresses this problem – the Cornell Knowledge Broker. It will not only support knowledge discovery, but also security, privacy, information exchange, and collaboration.

1 Introduction

A major challenge for intelligence agencies is to collect the relevant data. However, often an even greater challenge is to *analyze* that data and to draw conclusions from it. With increasing efforts for collecting more information, the data analysis problem will be aggravated. For that reason the Knowledge Discovery and Dissemination Working Group (KD-D) has initiated research on an information infrastructure that will meet the needs of U.S. intelligence and law enforcement agencies. This infrastructure will integrate new technology for online analysis of incoming data streams as well as novel offline data mining approaches. Based on discussions and projects within KD-D, we proposed the *information spheres* architecture to address both operational and legal requirements for intelligence agencies [18]. The architecture consists of two components – *local* and *global* information sphere.

A *local* information sphere exists within each government agency. In practice it could correspond to even smaller units, depending on restrictions of access to data. Its main goal is to support online analysis and data mining of multiple high-speed data streams, with conceptually unrestricted local access to all data managed by the system. Notice that information does not flow freely, i.e., the local information sphere also enforces access control to ensure that an analyst can only access what she *needs to know*.

The *global* information sphere spans multiple government agencies, and mediates inter-agency collaboration. It addresses the sometimes conflicting requirements of allowing analysts from different government agencies to efficiently share

* The authors are supported by NSF grants CCF-0205452, IIS-0121175, and IIS-0084762. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the sponsors.

H. Chen et al. (Eds.): ISI 2004, LNCS 3073, pp. 146–162, 2004.

information, hypotheses and evidence without violating applicable laws regarding privacy and civil rights. Hence the research challenges with respect to the global information sphere extend beyond efficient analysis and access control to privacy preserving data mining and information integration [2, 22, 27, 36, 43].

We propose the Cornell Knowledge Broker (CKB) which supports the functionality required for both information spheres: continuous processing of high-speed data streams from a variety of sources. This includes on-line data mining and trigger evaluation, deep offline analysis of archived data, as well as supporting interactive data analysis and exploration for analysts. The CKB will include novel techniques which are currently developed as part of the KD-D initiative, and it will follow the *plug-and-play* paradigm in order to support easy integration of new data mining and analysis operators and in order to be easily extended with new functionality. Already existing components are an adaptive subscription matcher, and a calculus for expressive subscriptions that can be stateful (across several documents, both temporally as well as content-wise). This will enable analysts to set up powerful filters for delivering relevant information from streams of incoming data in real-time. We are currently expanding the existing infrastructure to release a first version of the system in summer 2004.

2 System Architecture Overview

Figure 1 shows an overview of the information sphere architecture. Several local information spheres are connected with each other through a network. To preserve the privacy of individuals and also the privacy of the analyst (queries posed by an analyst might reveal that analyst's knowledge), access to a local sphere from the outside is restricted to privacy preserving technology. This technology is located in the Gateway Modules (GW). It allows analysts to leverage global knowledge within the bounds set by legal and security considerations.

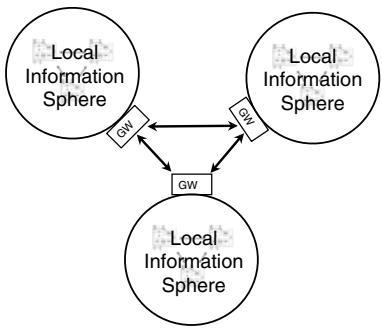

Fig. 1. Information sphere architecture

The local information sphere (cf. Figure 2) consists of one or more Cornell Knowledge Brokers, connected by a high-speed internal network (intranet). Each CKB has one or more processors with large main memory and essentially unrestricted archival capabilities, e.g., large hard disk drives or RAID arrays. We

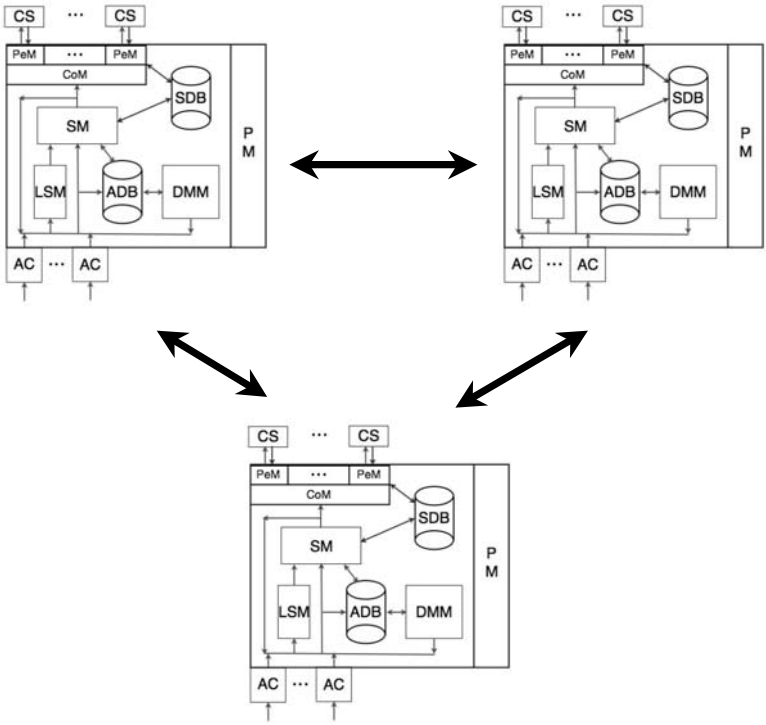

Fig. 2. Local information sphere

do not advocate any specific technology, e.g., a CKB unit could be a massively parallel machine or a cluster of PCs. Within a local information sphere information conceptually is allowed to flow freely. This does not imply that anybody can access any information. In fact, as we discuss later on, each CKB will support sophisticated access control mechanisms. The main difference from the global information sphere is that within a local information sphere data can essentially be stored and processed on any of the CKBs (by authorized users), requiring simpler interfaces that can be optimized for efficient information sharing. Stated differently, a user within the local information sphere can *trust* the system to enforce all applicable access control and security policies. Systems outside this trust boundary can only be accessed through the Gateway Modules. The Cornell Knowledge Broker will be discussed in detail in the next section.

3 The Cornell Knowledge Broker

The Cornell Knowledge Broker (see Figure 3) processes potentially massive streams of data, e.g., newsfeed data and continuously arriving measurements from sensors, but also reports from analysts, and other intelligence feeds. It supports on-line extraction of relevant information from streams in real-time, and

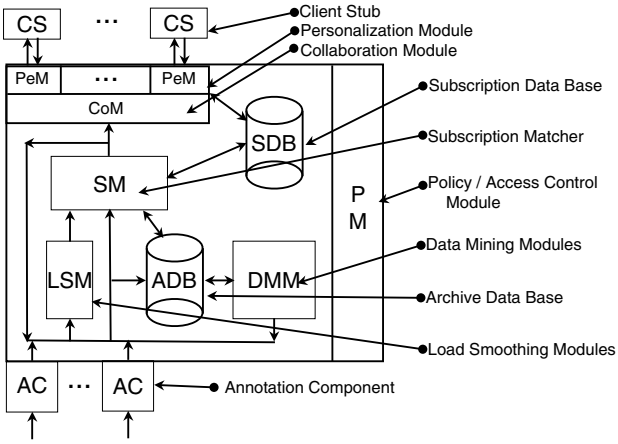

Fig. 3. Cornell Knowledge Broker

at the same time archives all incoming information for later deep offline analysis. Hence the CKB will ultimately combine the functionality of data warehouses [34] with that of data stream management systems (STREAM [5, 7], Aurora [12], TelegraphCQ [15], NiagaraCQ [16]) and sensor database systems [9, 37]. In the following sections we describe the components of the Cornell Knowledge Broker in more detail.

3.1 Annotation Component

The raw data that arrives at the knowledge broker might come in a variety of formats, e.g., streams of text data, relational data, images, voice recordings, sensor signals, and so on. Annotation Components (AC) transform this data into a coherent format which is amenable to data mining and stream processing techniques. For instance, an annotation module might parse incoming voice recordings and output the corresponding text. Another annotation module's task might be to analyze a plain text message and derive structural information from it like the topic of the text. Annotation modules hence add metadata to an item. This metadata will also describe in a uniform format where, when, and how this information was obtained or derived [10, 11].

In general we envision the annotation modules to annotate incoming data streams with XML metadata. XML is the ideal choice, because the majority of the data will be semi- or unstructured, typically plain text. Notice that this does not necessarily imply that the data be stored in native XML databases. Also, a mining tool might decide to first extract structured information from an XML document (certain attributes of interest) and then work on the structured information only.

3.2 Archiving Components

All incoming (annotated) information is stored for future reference and deep analysis in an Archive DataBase (ADB), shown in more detail in Figure 4. To support this functionality the CKB can initially rely on proven data warehousing technology. Data warehouses support sophisticated index structures and precomputed summaries for efficiently answering complex analysis queries over very large data sets. The major disadvantage is that updates cannot be applied in real-time since they would interfere with the query processing and considerably slow it down. Hence the archive will use a combination of an online DataBase (DB) together with a Data Warehouse (DW). The (much smaller) online database keeps track of newly arriving data in real-time, but is typically not accessible for queries. The updates collected in the online database are then periodically moved to the data warehouse in large batches, where they can be accessed by the data mining modules.

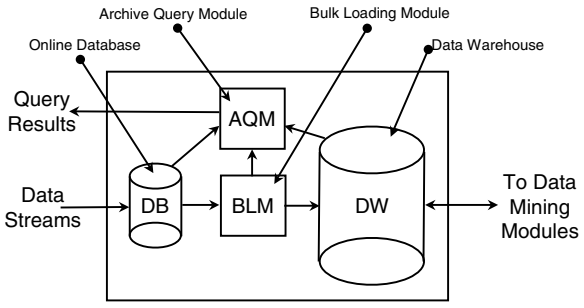

Fig. 4. Archive components

The Bulk Loading Module (BLM) controls the transfer of recent data from DB to DW. This setup is similar to current business architectures consisting of OLTP databases and data warehouses. BLM is fully aware of which data resides in DB and which has been moved to DW. It exports these statistics to the Archive Query Module (AQM), which handles queries from the Subscription Matcher (SM) and provides a consistent view of the archive database.

3.3 Data Mining Modules

Data Mining Modules (DMM) access the Data Warehouse in the archive to perform deep analysis over the collected information. Any technique for mining text data, images, numerical data, and so on, could be plugged in here. Data mining results need to be updated whenever a sufficient amount of new information has been moved from DB to DW. This update could be performed incrementally, or the data mining application computes a new result from scratch.

Within the context of the CKB the data mining modules are not just data consumers. They are also significant producers of new (derived) information.

The results produced by a DMM are fed back into the data processing life cycle within a CKB. In our current architecture data mining modules either generate (annotated) events or they populate relations (materialized views) in the archive. If events are generated, they can be processed in the Subscription Matcher like newly arriving information. The materialized views are available as input for other data mining applications.

The system can take advantage of this data cycle by incrementally extracting more detailed information from unstructured data, e.g., text and images, in a demand-driven manner. For example, assume in the beginning the metadata of a text consists only of the topic. In a first round of processing, a data mining tool determines the main keywords and adds them to the metadata. Based on this newly added information, the text could be labeled as "interesting" by an analyst's subscription (see discussion on Subscription Matcher below). This triggers another data mining application in the second round of processing to extract more detailed information with respect to some of the keywords, and so on.

The data analysis life cycle opens up several challenging research problems. First, how should different data mining applications be designed such that there is a maximum use of common intermediate results? Second, which of the intermediate results should be stored in the archive for later use? This problem is related to materialized view maintenance in data warehouses [29]. Third, how can events produced by diverse mining applications be efficiently processed such that large numbers of queries and high data arrival rates are supported?

3.4 Subscription Engine

The Subscription Matcher (SM) is part of the data processing cycle in the CKB. Its purpose is to analyze incoming data streams in real-time to alert analysts about relevant information and feed important data to data mining applications. For instance an analyst might set up a subscription that generates an alert whenever a document referring to a certain location or event is arriving. This would be a fairly simple subscription (filter predicate that compares an attribute value of incoming data with a given constant), like the ones supported by current publish/subscribe (pub/sub) systems [25].

For intelligence applications support for more complex subscriptions is required. This includes subscriptions concerning multiple events from multiple streams and temporal queries, e.g., discovery of temporal patterns. Similar to data mining applications (see Section 3.3), subscriptions can produce new events which are fed back into the system. For instance, there could be multiple low-level subscriptions that search for simple patterns in the incoming data. These patterns are then analyzed by more high-level subscriptions which discover more complex patterns consisting of simple patterns, and so on. Hence the subscription engine has to support *composite events*, i.e., queries can refer to events generated by other queries. Subscriptions are maintained in the Subscription DataBase (SDB).

Subscriptions may be defined such that their evaluation requires access to the archive. The main difference from data mining applications is the *real-time* requirement for processing subscriptions. The results of data mining applications could have considerable delay depending on the complexity of the algorithms.

Notice that there is an interesting duality between subscriptions and data mining. Subscriptions can be automatically generated by a data mining application, e.g., a mining workflow might have arrived at a stage where finding a new document mentioning A would result in a different next step than finding a document related to B. Hence the mining application would automatically generate a subscription to look for documents related to A or B. On arrival of a corresponding document, the mining application would be automatically notified and hence could continue its analysis appropriately. Similarly, a subscription set up by an analyst might trigger a deep analysis of historical information related to currently arriving data.

3.5 Load Smoothing

The Load Smoothing Module (LSM) ensures that the CKB does not collapse under sudden load spikes. As soon as LSM detects a possible overload situation, it pre-emptively eliminates some events from the data streams. Notice that the information is not lost, but its processing is deferred until the load on the system decreases.

LSM essentially works in two different modes. During load spikes it selects events to be removed from the system. Hence during that time only approximate results can be computed by those queries which are affected by dropped events. The goal of LSM during load spikes is to minimize given loss metrics for the currently active queries. Later, when the load returns to "normal" levels, LSM post-processes the initial approximate results until the exact results are obtained for all queries. As long as the system *in the average* has enough capacity to process incoming data, LSM smoothes load spikes by deferring computation.

The Load Smoothing Module addresses two challenging problems. First, it has to select which events to eliminate in order to cause the least loss in overall query accuracy during a load spike. Second, it has to decide which events to store in a fast database such that as soon as the load decreases, the exact result can be recovered. This is non-trivial, e.g., for joins not only the dropped events, but also matching partner-events need to be available for post-processing.

3.6 Personalization and Collaboration

The components related to personalization and collaboration are the SDB database, the Collaboration Module (CoM), and the Personalization Modules (PeM). The SDB database stores the personal profiles of analysts and applications. Here a personal profile contains metadata about current and past projects, keywords describing previous experience and expert knowledge, and of course the currently active subscriptions. This information affects the exchange of knowledge with other experts, but also between data mining applications. For instance, the

semantics of a subscription or the importance of incoming data might be different depending on an analyst's background and her current projects and active subscriptions and data mining workflows.

Similarly, Personalization Modules (PeM) will support on-line re-organization and annotation of incoming data to take personalization information of the data consumer into account. For example, a PeM can automatically learn search preferences of an analyst for providing customized rankings of large query result sets (see also Section 4.4).

One of the most challenging modules to develop is CoM. Its main goal is to *automatically* enhance the collaboration between analysts. This includes meta-computing, meaning that CoM will search for commonalities in the different subscriptions and data mining workflows. Assume two analysts are working on different cases, but there is a connection between these cases. By identifying commonalities in their analysis setup, CoM would be able to alert the analysts. To simplify information exchange, CoM would generate a description of the encountered similarities and some of the most relevant differences (e.g., data one analyst was not aware of).

The second functionality of CoM is to support *introspection*, i.e., "mining for best practice." CoM keeps track of previous cases by storing in the SDB database which approaches had success in the past and which were not as effective (e.g., which mining models and parameters were selected, which subscriptions were used, which data sources were combined in which way). If in the future an analyst works on a problem which has a similar structure as a past one, CoM could recommend "good" approaches or even perform an initial automatic setup of the required data mining workflow and subscriptions.

3.7 Policy and Access Control

The Policy and Access Control (PM) module ensures that only authorized users have access to certain data, including subscriptions and personalization related information. PM also contains some of the privacy-preserving functionality needed for information exchange with other participants in a global information sphere. The functionality of PM therefore extends far beyond simple access control in operating systems, e.g., Windows or UNIX file ownership attributes. There has been work on security and access control for statistical and XML databases [1, 8, 45], but to date none of the proposed techniques provides a satisfactory combination of guaranteeing access restrictions while supporting desired analysis functionality.

We identified the following key features for PM. First, PM should support *fine-grained* access control, down to the level of single attribute values in documents. This results in potentially high overhead for enforcing policies, but will enable an analyst to access that and only that data that she needs to know. Second, there should be support for *dynamic* policies, i.e., the overhead for changing policies should be low. For instance, assume during an investigation an analyst detects a link between two documents in the database. Both documents are accessible to the analyst, but a third document which forms the link between the

two currently is not. Depending on the case it might be necessary to extend the analyst's need-to-know in order to let her proceed with the investigation. Third, PM should support *adaptive content-based* access control. When new documents arrive at the system, it might not be initially clear which access policies apply. Only after several processing steps, when a document's content has been sufficiently analyzed, can proper access rights be assigned. In order to take as much of the computational burden away from human operators, PM should be able to automatically update access rights as the document is processed.

4 Current Status and Next Steps

We have available several techniques for the different modules, mostly for the Subscription Matcher, Load Smoothing and Data Mining Modules. In the following sections some of these techniques are surveyed, accompanied by a discussion how the different components will be connected in an initial version of the Cornell Knowledge Broker.

4.1 Techniques for the Subscription Matcher

The purpose of the Subscription Matcher is to analyze incoming data streams in real time to alert analysts about relevant information and feed important data to data mining applications. It is functionally similar to a continuous data stream query processor [5, 7, 12, 15, 16], a publish/subscribe system [25], or an event processing system [3, 13, 14, 38, 44]. However, there are significant differences.

In a conventional pub/sub system the query language typically consists of simple filter predicates–Boolean expressions evaluated on a single record of the publication stream [25]. Such queries lend themselves to efficient implementation. However, while they are able to select individual records or events of interest, they are obviously not able to identify spatial or temporal patterns, which involve multiple events, possibly from different input streams. We believe supporting such patterns is essential for intelligence applications. Thus, a more expressive subscription query language is required.

One very powerful approach is represented by the CQL Continuous Query Language implemented in the Stanford STREAM system [4]. In STREAM, data is modeled both as timestamped streams and as conventional relations. CQL includes the full SQL language, and performs the bulk of its data manipulation in the relational domain. In addition, there is a small set of stream-to-relation and relation-to-stream operators, used to translate stream data and query results between the stream and relational domains. While formally appealing, this approach has the drawback that it can easily express stream queries that are infeasibly expensive to execute; and there are no obvious criteria the user can apply to ensure her queries will be efficient.

We are taking a different approach. Instead of defining a single very powerful data stream query language, we are developing a *hierarchy* of query languages. At the bottom is a data stream query language that we already know how to implement very efficiently – simple attribute/value based publish/subscribe. Moving up in the hierarchy yields increasingly more expressive languages, with

increasingly more expensive query execution. Instead of targeting one design point whose query language has great expressiveness, we are developing a "sliding scale" of expressiveness/performance design points which will allow us to control the tradeoff between expressiveness and performance.

The languages in our hierarchy are capable of defining "composite events" as discussed in Section 3.4. Query results produce events whose treatment by subsequent queries is indistinguishable from the treatment of primitive events arriving on the input data streams.

We are developing a general calculus for data stream queries. This calculus will provide the means for comparing the expressiveness and complexity of different query languages in a precise, formal way. Our preliminary results are quite encouraging, and we believe they validate our fundamental design choices:

- A data model comprising temporally-annotated data streams;
- A two-sorted first order logic including data terms and temporal terms, with strict limitations on the use of temporal terms;
- A specialized binding construct to allow parameterized aggregation.

Our calculus is powerful enough to simulate the majority of previous work in languages for data stream processing systems, pub/sub, and event processing systems. At the same time, we have been able to find fairly simple syntactic characterizations of subsystems that are semantically equivalent to the query systems we have examined, even systems that are much less expressive than our full calculus. The characterizations use properties such as nesting depth of operators and quantifiers, and the number of free variables in a formula – properties that intuitively have a natural relation to execution complexity, and in some cases translate directly to the structure of a message-oriented implementation of the subscription engine.

4.2 Techniques for the Data Mining Module

In the following we survey techniques that will be integrated as part of the Data Mining Module. Other data mining approaches can be easily added later on as the CKB evolves.

Change Detection. One of the major challenges presented by data streams is detecting a change. Up to now, work in this area has taken a parametric approach: the data has been modeled by prespecified families of distributions or Markov processes. A change is announced whenever an algorithm finds a model in this family that describes the current data better than the current model. We are working on a completely different, *nonparametric* approach. We make no assumptions about the distribution of data except that the records are generated independently; that is, the value of one record does not depend on the values of the records that appeared before it in the data stream. This technique has several advantages: it is intuitive, it is completely general (it is independent of the distributions that generate the data), works for discrete and continuous data, and shows good results experimentally.

Change has far-reaching impact on any data processing algorithm. For example, when constructing a data mining model over a data stream, old data before a change can bias a data mining model towards data characteristics that do not hold any longer. If we process queries over data streams, we may want to give separate query answers for each time interval where the underlying data distribution is stable. Our work is based on a formal definition of change and associated techniques for quickly detecting whenever statistically significant change has occurred.

Hopcroft et al. attack this problem for graph-structured data [30]. They examine how to detect communities in large linked networks, and how to track such communities over time. They focus in particular on data with a low signal-to-noise ratio.

Detecting Bursts in Data Streams. Kleinberg [35] proposes a framework for modeling "bursts" in temporal data, in such a way that they can be robustly and efficiently identified, and can provide an organizational framework for analyzing the underlying content. Many of the most widely used on-line media and Web information sources can be naturally viewed as continuous streams of information. Their time scales range from the minute-by-minute dynamics of usage at high-volume Web sites, to the hourly and daily evolution of topics in e-mail, news, and blogs, to the long-term research trends that are evident in research paper archives. Kleinberg's approach rests on the premise that the appearance of a topic or event is signaled by a "burst of activity," with certain features rising sharply in frequency as the topic emerges. The approach is based on modeling the stream using an infinite-state automaton, in which bursts appear as state transitions. The automaton also implicitly imposes a hierarchical structure on the bursts in a stream.

4.3 Techniques for the Load Smoothing Module

As discussed in Section 3.5, load smoothing consists of two different modes of operation: peak-load mode and low-load mode. During peak load, LSM removes events from the system in order to avoid thrashing. This is also referred to as *load shedding*. Most current approaches are based on random load shedding [6, 33, 42] (Tatbul et al. [42] also consider simple heuristics for semantic load shedding, where certain events have higher value to the query than others). Random load shedding works well for queries that compute aggregates, but as we show [17], random load shedding is inferior if we are concerned with the approximation quality of set-valued query results. In intelligence applications we expect a large fraction of the queries to have set-valued results, e.g., queries searching for certain documents or groups of similar documents from multiple input streams. Notice that we treat incoming documents as events with attached application data.

In [17] we propose novel approximation techniques for joins of data streams, i.e., where we are interested in finding matching (similar) documents or events in different data streams. We examine several popular and established approximation measures for sets. In our load shedding scenario most of these measures

reduce to the MAX-subset measure: the best approximate result is the one which is the largest subset of the exact query answer. For MAX-subset we develop an optimal *offline* algorithm. For given input sets it determines in polynomial time (polynomial in parameters like input size, memory size, and size of the join window for sliding window joins [17]) the optimal strategy of keeping or dropping incoming events. This algorithm assumes complete future knowledge and hence is not applicable in a real system. However, it serves as a benchmark for comparing *any* practical online strategy with what is the best any such strategy could ever achieve with the given system resources.

As it turns out, as long as one can approximately predict future arrival probabilities of similar events, simple random load shedding is far inferior to semantic join approximation techniques like PROB [17]. PROB is a simple lightweight heuristic that in case of system overload drops the events with the lowest predicted arrival probability of a matching partner event in the other stream. PROB will therefore be added to LSM.

Unfortunately the optimization problem of minimizing result loss during peak load periods is very complex in general. To date there is no complete end-to-end load shedding approach except for simple random load shedding. The initial version of CKB therefore will support semantic join approximation as well as random load shedding for aggregation queries.

The second mode of operation of LSM, the low-load mode, presents new challenges which have not been sufficiently examined by the database community. In [17] we introduce the Archive-Metric (ArM) which measures the post-processing cost for the recovery of the exact result after a load spike. We have developed system models for optimizing different cost metrics related to accessing and processing all events that were dropped during peak load periods. Unfortunately optimal offline algorithms for this problem so far are too complex and hence not useful for benchmarking. We are currently developing efficient online heuristics that balance parameters like archive access cost, result latency, and total processing cost in terms of CPU and memory usage.

4.4 Techniques for Personalization and Collaboration

In recent work Joachims [32] proposes a novel approach to optimizing retrieval functions using implicit feedback. The main idea is to automatically learn functions that rank documents according to their relevance to a query. Taking an empirical-risk-minimization approach with Kendall's Tau as the loss function, a Support Vector algorithm is defined for the resulting convex training problem. An important property of this algorithm is that it can be trained with partial information about the target rankings like "for query Q, document A should be ranked higher than document B". In Web search applications such preference data is available in abundance, since it can be inferred from the clicking behavior of users. We expect similar properties to hold for intelligence applications, allowing the system to learn an analyst's preferences with respect to a certain query or project.

4.5 Privacy-Preserving Data Mining

As outlined in Section 1, a vital component of our system will be the capability to perform privacy-preserving computation and data mining in the global information sphere. Assume that there is an analyst in one local information sphere that initiates the privacy-preserving computation, and we call this information sphere the *server sphere*. We call the information spheres that participate in the computation by engaging in a protocol for privacy-preserving computation *client spheres*. For simplicity, we assume that there is one server and a set of clients, each having some data.

The usual solution to the above problem consists in having all client peers send their private data to the server sphere. However, the system that we are building will be able to accomplish this computation *without violating the privacy of individual clients*. In other words, if all the server needs is a data mining model, a solution is preferred that reduces the disclosure of private data while still allowing the server to build the model. Similarly, if all the server needs is the answer to an aggregate query, the computation of this query should not disclose anything beyond the answer to the query (and what can be concluded from the query answer).

One possibility is as follows: before sending its piece of data, each client sphere perturbs it so that some true information is taken away and some false information is introduced. This approach is called *randomization*. Another possibility is to decrease precision of the transmitted data by rounding, suppressing certain values, replacing values with intervals, or replacing categorical values by more general categories up the taxonomic hierarchy, see [19, 31, 40, 41].

The usage of randomization for preserving privacy has been studied extensively in the framework of statistical databases [20, 21, 26, 28, 39]. In that case, the server has a complete and precise database with the information from its clients, and it has to make a version of this database public, for others to work with. One important example is census data: the government of a country collects private information about its inhabitants, and then has to turn this data into a tool for research and economic planning. However, it is assumed that private records of any given person should not be released nor be recoverable from what is released. In particular, a company should not be able to match up records in the publicly released database with the corresponding records in the company's own database of its customers.

Our recent work in this area has addressed the problem of privacy-preserving association rule mining, and we introduced a novel notion of *privacy breaches*, a formal definition of how much privacy is compromised for a given privacy-preserving data mining method [24, 23].

4.6 How to Connect the Different Modules

At first glance, implementing the infrastructure that connects the different modules of the CKB might appear to be a substantial effort. We believe this is not the case. While the CKB's infrastructure does have some unusual requirements,

it is fundamentally similar to many high-performance enterprise applications. The requirements are generally quite close to what is provided in existing J2EE implementations.

We propose to start from an existing open-source J2EE system such as JBoss (`http://sourceforge.net/projects/jboss`). Rather than building new infrastructure, we view ourselves as building a medium-size J2EE application, possibly with minor enhancements to a few of the APIs – e.g., enhancing JMS to support our richer subscription semantics. Developing a CKB prototype in this way should be well within the capabilities of a university research group. Stated differently, by relying on J2EE technology we can concentrate our efforts on the design of the CKB modules and their interaction.

5 Conclusion

The major challenge for intelligence applications is to find relevant information in massive amounts of available data, both data which is already archived as well as newly arriving data. We have presented the Cornell Knowledge Broker, which constitutes the information processing unit of the information sphere architecture. The CKB takes a unique approach to combining offline deep analysis (traditional data mining) with real-time online processing, while at the same time providing support for access control and privacy-preserving technology.

We do not claim that the Cornell Knowledge Broker and the Information System architecture are the only possible way to support analysts in processing massive amounts of data. Indeed, the architecture itself leaves room for alternative implementations. Our main goal is to create an infrastructure that integrates novel data mining techniques to maximize their effectiveness.

Our future work will therefore follow two directions. First, we will continue to develop novel techniques for mining large data sets and data streams. These techniques lie at the heart of the actual data processing task performed by an analyst. Second, we will extend the system architecture to integrate these techniques to maximize their effectiveness, enabling them to interact with each other and with system services such as access control. We are currently implementing several of the CKB's modules and expect a first prototype of the system to be ready in the second half of 2004. Once a prototype system is available, we will be able to evaluate our architecture design decisions and to make quantitative comparisons to alternative approaches.

References

1. N.R. Adam and J.C. Wortmann. Security-control methods for statistical databases: A comparative study. *ACM Computing Surveys*, 21(4):515–556, 1989.
2. R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 439–450, 2000.
3. M. K. Aguilera, R. E. Strom, D. C. Sturman, M. Astley, and T. D. Chandra. Matching events in a content-based subscription system. In *Proc. ACM Symp. on Principles of Distributed Computing (PODC)*, pages 53–61, 1999.

4. A. Arasu, S. Babu, and J. Widom. The CQL continuous query language: Semantic foundations and query execution. Technical report, Stanford University, 2003.
5. B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *Proc. Symp. on Principles of Database Systems (PODS)*, pages 1–16, 2002.
6. B. Babcock, M. Datar, and R. Motwani. Load shedding techniques for data stream systems (short paper). In *Proc. Workshop on Management and Processing of Data Streams (MPDS)*, 2003.
7. S. Babu and J. Widom. Continuous queries over data streams. *ACM SIGMOD Record*, 30(3):109–120, 2001.
8. E. Bertino, S. Jajodia, and P. Samarati. Database security: Research and practice. *Information Systems*, 20(7):537–556, 1995.
9. P. Bonnet, J. Gehrke, and P. Seshadri. Towards sensor database systems. In *Proc. Int. Conf. on Mobile Data Management (MDM)*, pages 3–14, 2001.
10. P. Buneman, S. Khanna, K. Tajima, and W. C. Tan. Archiving scientific data. In *Proc. SIGMOD*, pages 1–12, 2002.
11. P. Buneman, S. Khanna, and W. C. Tan. Why and where: A characterization of data provenance. In *Proc. Int. Conf. on Database Theory (ICDT)*, pages 316–330, 2001.
12. D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, G. Seidman, M. Stonebraker, N. Tatbul, and S. Zdonik. Monitoring streams – a new class of data management applications. In *Proc. Int. Conf. on Very Large Databases (VLDB)*, 2002.
13. A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Achieving scalability and expressiveness in an internet-scale event notification service. In *Proc. ACM Symp. on Principles of Distributed Computing (PODC)*, pages 219–227, 2000.
14. S. Chakravarthy, V. Krishnaprasad, E. Anwar, and S.-K. Kim. Composite events for active databases: Semantics, contexts and detection. In *Proc. Int. Conf. on Very Large Databases (VLDB)*, pages 606–617, 1994.
15. S. Chandrasekaran, O. Cooper, A. Deshpande, M. J. Franklin, J. M. Hellerstein, W. Hong, S. Krishnamurthy, S. R. Madden, V. Raman, F. Reiss, and M. A. Shah. TelegraphCQ: Continuous dataflow processing for an uncertain world. In *Proc. Conf. on Innovative Data Systems Research (CIDR)*, 2003.
16. J. Chen, D. J. DeWitt, F. Tian, and Y. Wang. NiagaraCQ: A scalable continuous query system for internet databases. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 379–390, 2000.
17. A. Das, J. Gehrke, and M. Riedewald. Approximate join processing over data streams. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 40–51, 2003.
18. A. Demers, J. Gehrke, and M. Riedewald. Research issues in mining and monitoring of intelligence data. In H. Kargupta, A. Joshi, K. Sivakumar, and Y. Yesha, editors, *Data Mining: Next Generation Challenges and Future Directions*. MIT/AAAI Press, 2004. To be released.
19. George T. Duncan, Ramayya Krishnan, Rema Padman, Phyllis Reuther, and Stephen Roehrig. Cell suppression to limit content-based disclosure. In *Proceedings of the 30th Hawaii International Conference on System Sciences*, volume 3. IEEE Computer Society Press, 1997.
20. George T. Duncan and Sumitra Mukherjee. Optimal disclosure limitation strategy in statistical databases: Deterring tracker attacks through additive noise. *Journal of the American Statistical Association*, 95(451):720–729, 2000.

21. Timothy Evans, Laura Zayatz, and John Slanta. Using noise for disclosure limitation of establishment tabular data. *Journal of Official Statistics*, 14(4):537–551, 1998.
22. A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, 2002.
23. Alexandre Evfimievski, Johannes Gehrke, and Ramakrishnan Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proceedings of the 22nd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS 2003)*, 2003.
24. Alexandre Evfimievski, Ramakrishnan Srikant, Rakesh Agrawal, and Johannes Gehrke. Privacy preserving mining of association rules. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery in Databases and Data Mining*, pages 217–228, Edmonton, Alberta, Canada, July 23–26 2002.
25. F. Fabret, H.-A. Jacobsen, F. Llirbat, J. Pereira, K. A. Ross, and D. Shasha. Filtering algorithms and implementation for very fast publish/subscribe. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 115–126, 2001.
26. Stephen E. Fienberg, Udi E. Makov, and Russel J. Steele. Disclosure limitation using perturbation and related methods for categorical data. *Journal of Official Statistics*, 14(4):485–502, 1998.
27. J. Gehrke, editor. *Special Issue on Privacy and Security*, volume 4 of *SIGKDD Explorations*, 2002.
28. J.M̃. Gouweleeuw, P. Kooiman, L. C. R.J̃. Willenborg, and P.-P. de Wolf. Post randomisation for statistical disclosure control: Theory and implementation. *Journal of Official Statistics*, 14(4):463–478, 1998.
29. A. Gupta and I. S. Mumick, editors. *Materialized Views: Techniques, Implementations, and Applications*. MIT Press, 1998.
30. J. E. Hopcroft, O. Khan, B. Kulis, and B. Selman. Natural communities in large linked networks. In *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 541–546, 2003.
31. Vijay S. Iyengar. Transforming data to satisfy privacy constraints. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery in Databases and Data Mining*, pages 279–288, Edmonton, Alberta, Canada, July 23–26 2002.
32. T. Joachims. Optimizing search engines using clickthrough data. In *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 133–142, 2002.
33. J. Kang, J. F. Naughton, and S. D. Viglas. Evaluating window joins over unbounded streams. In *Proc. Int. Conf. on Data Engineering (ICDE)*, 2003.
34. R. Kimball. *The Data Warehouse Toolkit*. John Wiley and Sons, 1996.
35. J. M. Kleinberg. Bursty and hierarchical structure in streams. *Data Mining and Knowledge Discovery*, 7(4):373–397, 2003.
36. Y. Lindell and B. Pinkas. Privacy preserving data mining. *Journal of Cryptology*, 15(3):177–206, 2002.
37. S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. The design of an acquisitional query processor for sensor networks. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 491–502, 2003.
38. I. Motakis and C. Zaniolo. Formal semantics for composite temporal events in active database rules. *Journal of Systems Integration*, 7(3-4):291–325, 1997.

39. Sumitra Mukherjee and George T. Duncan. Disclosure limitation through additive noise data masking: Analysis of skewed sensitive data. In *Proceedings of the 30th Hawaii International Conference on System Sciences*, volume 3, pages 581–586. IEEE Computer Society Press, 1997.
40. Pierangela Samarati and Latanya Sweeney. Generalizing data to provide anonymity when disclosing information. In *Proceedings of the 17th ACM Symposium on Principles of Database Systems*, Seattle, Washington, USA, June 1–3 1998.
41. Pierangela Samarati and Latanya Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, Oakland, California, USA, May 1998.
42. N. Tatbul, U. Çetintemel, S. Zdonik, Mitch Cherniack, and Michael Stonebraker. Load shedding in a data stream manager. In *Proc. Int. Conf. on Very Large Databases (VLDB)*, pages 309–320, 2003.
43. J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 23–26, 2002.
44. A. Yalamanchi, J. Srinivasan, and D. Gawlick. Managing expressions as data in relational database systems. In *Proc. Conf. on Innovative Data Systems Research (CIDR)*, 2003.
45. T. Yu, D. Srivastava, L. V. S. Lakshmanan, and H. V. Jagadish. Compressed accessibility map: Efficient access control for xml. In *Proc. VLDB*, pages 478–489, 2002.