

# Managing Disclosure of Private Financial Data with Hippocratic Databases

Rakesh Agrawal, Dmitri Asonov, Roberto Bayardo,  
Tyrone Grandison, Christopher Johnson

*Intelligent Information Systems Group,  
IBM Almaden Research Center  
San Jose, CA 95120, USA.*

*{ragrawal, dasonov, bayardo, tyroneg, johnsocm}@us.ibm.com*

**WHITE PAPER**

January 2005

## Table of Contents

1	Executive Summary .....	2
2	Functionality .....	4
2.1	Active Enforcement .....	4
2.2	Compliance Auditing .....	6
2.3	Sovereign Information Integration.....	8
3	Data Warehouse Use Case Scenario .....	10
3.1	Carnegie Astor & Co. ....	10
3.2	Problem .....	10
3.3	New Credit Card Customer - Claire.....	11
3.4	CMS Mortgage Broker - Brian .....	11
3.4.1	Results of Query without HDB Active Enforcement .....	11
3.4.2	Results of Query with HDB Active Enforcement.....	12
3.5	Compliance Verification Using Eunomia Auditing.....	12
3.5.1	Claire’s Complaint.....	12
3.5.2	Investigative Audit.....	13
3.5.3	Audit Report.....	14
3.5.4	Post-Audit Investigation .....	14
3.6	Sharing Information with SII .....	15
4	Technical Discussion .....	17
4.1	HDB Active Enforcement.....	17
4.1.1	Policy Installation .....	17
4.1.2	User Preferences and Data Collection .....	17
4.1.3	Query Enforcement.....	18
4.1.4	Component Assumptions .....	18
4.1.5	Performance .....	19
4.2	HDB Eunomia Compliance Auditing .....	21
4.2.1	Logical Logging.....	21
4.2.2	HDB Audit Application .....	22
4.2.3	Component Assumptions .....	23
4.2.4	Performance .....	23
4.3	HDB Sovereign Information Integration .....	25
4.3.1	Architecture.....	25
4.3.2	Performance .....	25
5	Conclusion .....	27
	References.....	30

## 1 Executive Summary

Companies in the financial services industry manage large volumes of sensitive customer data. These companies face increasing legal regulation and customer pressure to impose stringent privacy and security controls on access and disclosure of electronic customer records. If financial service companies do not take decisive action to protect this sensitive information, they risk exposure to costly government fines and customer lawsuits. They also risk losing customers to competitors with stronger privacy protection. IBM's Hippocratic Database (HDB) technology offers a scalable solution for automated management of large volumes of sensitive data. HDB's three components -- Active Enforcement, Compliance Auditing, and Sovereign Information Integration -- ensure that companies derive the maximum value from sensitive data without violating the law or otherwise compromising security or privacy.

HDB's Active Enforcement (AE) component automates enforcement by rewriting user queries in a layer above the database and returning only data that is consistent with company policies, applicable legislation, and customer preferences. The Compliance Auditing component records all queries and changes to the database and uses this information to construct detailed audit trails that specify the user, recipient, purpose, time, and exact (cell-level) information disclosed for any particular database query. Finally, the Sovereign Information Integration (SII) component allows two parties to share information about intersections between data sets without compromising the privacy or security of the remaining data.

HDB operates as a middleware layer between the database and enterprise applications. HDB offers strong advantages over competing solutions because it: (i) is transparent to enterprise applications and agnostic to database systems; (ii) uses the substantial computing power of the database, thus promoting scalability and performance; (iii) manages data access and disclosure down to the cell level of the database, while most solutions operate only at the row or column level; (iv) facilitates accurate and efficient compliance auditing by recording the minimum amount of information necessary to reconstruct the state of any cell in the database at any given

point in time; and (v) allows information sharing across autonomous data sources that provides the results of queries without revealing other data.

Part Two of this whitepaper describes the functionality of the three HDB components and their advantages over other methods of data disclosure management. Part Three outlines several financial services scenarios that demonstrate practical applications of the HDB. Part Four explains the technical architecture and analyzes the performance of each of the components. We note that while this paper outlines enforcement of privacy and security policies in a financial services context, HDB technology is extensible to a variety of industries and regulatory environments requiring policy-based disclosure management.

## 2 Functionality

HDB technology actively enforces data disclosure policy, allows secure sharing of information, and facilitates auditing and compliance with legislation (e.g., GLB, HIPAA, EU Privacy Directive, etc.). Its three functional components -- Active Enforcement, Compliance Auditing and Sovereign Information Integration -- provide distinct advantages to financial services companies in managing sensitive customer data. In this section, we describe each of these three components, highlighting their functionality and advantages over other solutions.

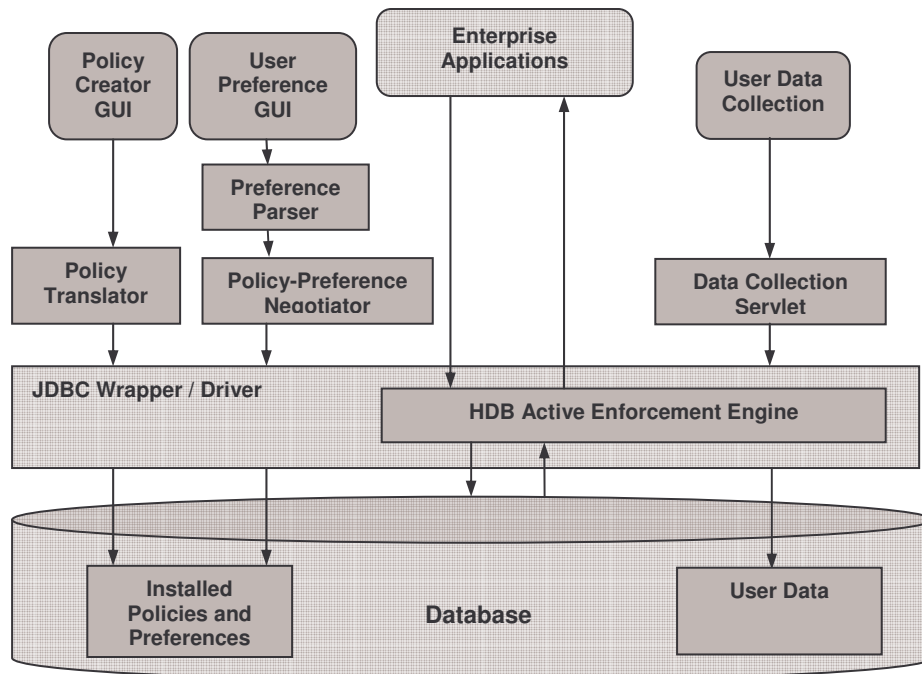
### 2.1 Active Enforcement

Active Enforcement [1,2] ensures that applications accessing a HDB-enabled data source adhere to data disclosure policies. (See Figure 1.) It allows applications to enforce disclosure policy on arbitrary data elements in an automated fashion. One of the decisive advantages of AE is its ability to manage disclosure at the cell level in the database, rather than just the row or the column level. AE includes the following three phases:

- **Policy Creation** involves the specification of the rules governing data disclosure. The company adopts a data disclosure policy to specify (i) who is allowed to access what information; and (ii) for what purposes information may be disclosed to each recipient. The company may develop more detailed data disclosure policies to comply with various regulatory requirements or company regulations. The policies are expressed in a privacy language such as W3C's Platform for Privacy Preferences (P3P) and installed in the AE engine in a form amenable to symbolic manipulation.
- **Preference Stipulation** is the process whereby a customer defines his or her personal preferences regarding information access and usage. This stage occurs prior to the customer providing any personal data to the financial institution. The customer indicates his preferences regarding the use and disclosure of his personal data in a privacy preference language [3]. HDB then matches these preferences with the company's privacy policies to identify any conflicts. The customer is advised of

these conflicts and given an opportunity to resolve them or terminate the relationship. The customer is then given a series of opt-in and opt-out choices that allow him to further refine his privacy preferences. A successful negotiation confirms agreement between the customer and company regarding disclosure of personal data.

- Application Data Retrieval** is the final phase of Active Enforcement. In this phase, all queries to be executed on the data source are programmatically modified so that the application only retrieves results that are compliant with company disclosure policies (including legal requirements) and user preferences (including opt-in and opt-out choices). This process is fully automated and is only dependent upon negotiated privacy policies and preferences.



*Figure 1: Hippocratic Database Active Enforcement Architecture*

Depending on the application domain and the nature of the problem, the **Preference Stipulation** component may be excluded. For instance, certain organizations may be responsible for enforcing privacy policies that have already been negotiated by an affiliate company and therefore have no need to negotiate privacy preferences.

The key strengths of AE are that: (i) it offers a general methodology for handling and codifying policy and preference information; (ii) its policy enforcement is transparent to enterprise applications (integration assumes a SQL interface such as ODBC or JDBC); (iii) in the typical case, it improves query processing speed; and (iv) it is agnostic to underlying relational database technology (AE can work with DB2, Oracle, Microsoft, or any other rich relational data source).

## 2.2 Compliance Auditing

HDB's Compliance Auditing [4] component, called Eunomia, enables companies to verify compliance with data disclosure laws, company policies, and customer preferences. Eunomia consists of two basic parts – a logical logging system and an audit application. The logical logging system records all query and context information (identity, purpose, time, etc) in query logs. It also stores all data updates, insertions, and deletions in backlog tables. The audit application uses the query logs and backlog tables to reconstruct the state of the database at any given time in the past. Upon receiving an audit expression from an auditor interface, the application uses a query generator to select candidate queries from the query logs. The application returns an audit trail that identifies the user, recipient, purpose, and time of candidate queries and the exact information disclosed by each query. (See Figure 2.)

There are two models of Eunomia – an in-house model and an outsourced model. Both provide the same basic functionality. The difference between the two lies in the methods of creating and storing the backlog tables and query logs. The in-house model uses triggers to create and store these tables within the database. In contrast, the outsourced model uses the database's recovery logs to construct backlog tables and query logs, which are maintained in a remote database. The outsourced model allows compliance auditing to be conducted by a remote service, rather than within the company. The query logs and backlog tables need only to be updated, in batch, with other maintenance tasks or when necessitated by an audit request.

Eunomia has significant advantages over other auditing techniques, such as query logging or result logging. Query logging involves recording only the queries themselves,

but not the data returned. The obvious disadvantage of this method is that it is able to audit only the requested data, but does not keep track of the actual data accessed or disclosed. On the other hand, result logging involves recording all records returned in response to every query. As described below, Eunomia's logical logging methodology has efficiency, storage, security, and usability advantages over result logging systems.

- **Efficiency:** Logical logging is much more efficient than result logging because it does not capture unnecessary data. "Read-only" queries, in which no changes are made to the database, comprise as much as 80 to 95% of the queries that access a typical database. Logical logging records the query itself, but none of the records accessed, for read-only queries. Research shows that the overall cost of logical logging is almost always significantly less than result logging.
- **Storage:** In enterprise environments, query results tend to be very large data sets and logging these results requires unusually high storage overhead. Eunomia avoids this problem by storing only inserts, updates, and deletions. This eliminates storing large quantities of redundant information, without affecting the ability to reconstruct the state of the database at any given point in time.
- **Security:** Eunomia also has a security advantage over database auditing systems that log only query results, as such systems may not track all information actually disclosed in response to particular queries. Query results often do not actually show the precise data disclosed in response to a query. For example, users can formulate queries to return nonsense data (e.g., *value X*) whenever the presence of sensitive data is detected. Such a query could be written as:

*Select [value X] if customer [Jane] has an annual income > \$100,000*

The output only reflects that *value X* was disclosed, when in reality, the database disclosed that Jane's annual income exceeds \$100,000.

For another example, result logging systems may allow users to determine information about particular customers by running a series of aggregate queries without tracking that any sensitive information was disclosed. In contrast, Eunomia tracks these types of unauthorized disclosures because it logs the queries themselves. In fact,

administrators can insert audit flags to alert them to such suspicious access.

- Ease of use:** Eunomia is also much easier for companies to use than other auditing solutions because it is transparent to existing database applications. The administrator configures the system through a simple wizard which sets up a small set of tables and triggers on the database. Applications connect to the proxy instead of directly to the database and users can continue to use the same SQL statements as before. The Eunomia audit application allows auditors to detect disclosures without having to understand the applications that access the database.

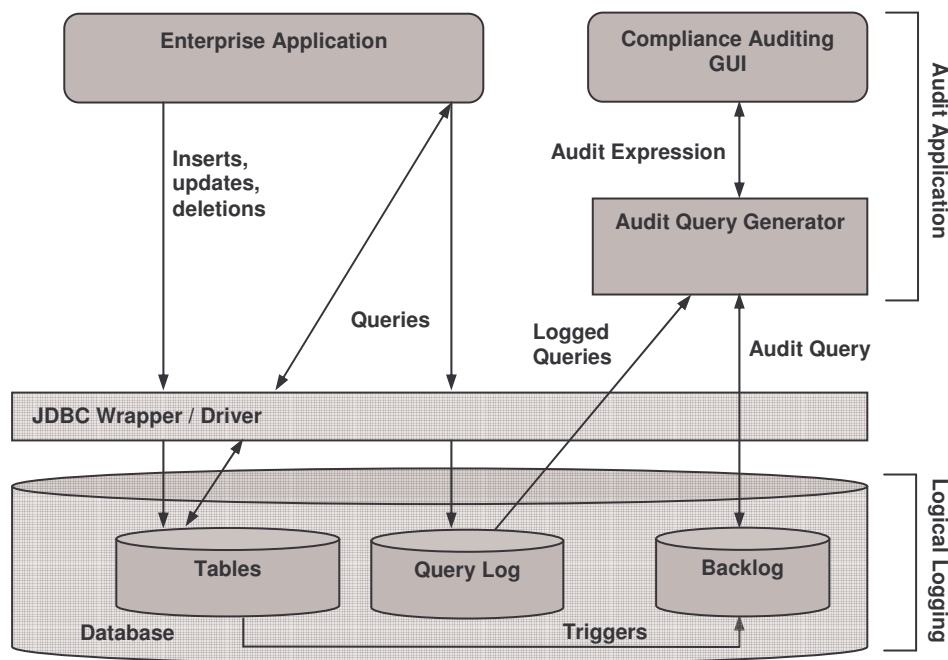


Figure 2: Eunomia Compliance Auditing (In-House Model)

### 2.3 Sovereign Information Integration

Sovereign Information Integration (SII) [5,6] enables two or more autonomous entities to compute queries across their databases in such way that only the results of the query are revealed. SII uses a web services infrastructure to apply a set of commutative encryption functions to uniquely identifiable data in different orders and at different locations. The resulting multiply encrypted values can then be compared without compromising the privacy or security of either data set. (See Figure 3.)

Unlike other data integration approaches, such as centralized data warehouses or mediator-based data federations, which reveal all data among the databases, SII does not reveal any information among the databases other than the results of the query to the calling entity. This allows a variety of joins and other operations to be performed across databases without revealing any confidential information.

SII is a scalable middleware solution that can be integrated seamlessly into existing data environments without the need for a trusted third party or any anonymization of the original data.

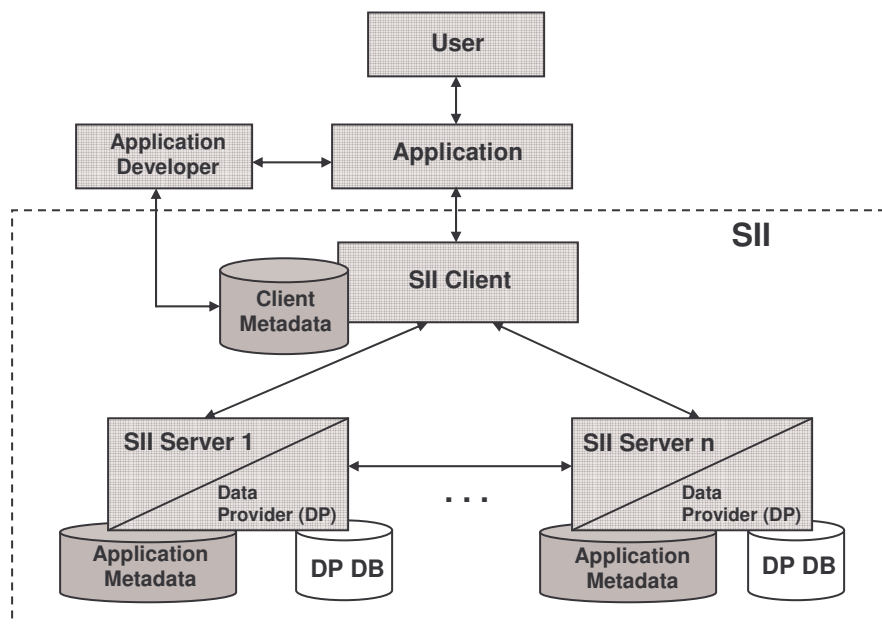


Figure 3: HDB Sovereign Information Integration (SII)

### 3 Data Warehouse Use Case Scenario

Financial service companies are required to ensure the privacy and security of large volumes of sensitive customer information in accordance with information disclosure laws, company privacy policies, and customer choices and preferences. This section describes a fictitious scenario involving the integration of databases between several companies under common ownership. The scenario illustrates the automation of disclosure controls, compliance auditing, and secure information sharing among these companies using HDB technology.

#### 3.1 Carnegie Astor & Co.

Carnegie Astor & Co. (“Carnegie”) is a fictional conglomerate of financial service companies resulting from the merger of the Carnegie Companies and JJ Astor Bank. Carnegie now includes the following companies:

- Carnegie Investment Bank
- Carnegie Mortgage Services
- JJ Astor Bank USA
- JJ Astor Insurance Company

#### 3.2 Problem

Carnegie is currently attempting to consolidate the databases of its various companies into a single data warehouse. The management of JJ Astor Bank USA (“Astor”) and Carnegie Mortgage Services (“CMS”) are concerned that the customer data will not be kept secure and private between the two companies. Astor has one of the largest credit card portfolios in the United States, while CMS has a substantial portfolio of residential real estate loans. Carnegie needs a solution that will allow it to consolidate diverse data sources without compromising privacy or security among its four business units. The following situation assumes that the database consolidation has already taken place and explores the extent of privacy protection with and without HDB technology.

### 3.3 New Credit Card Customer - Claire

Claire Young is a 31-year old professional in Santa Monica, California who would like to open a platinum credit card account with Astor. She logs onto Astor's website and completes an application, divulging a significant amount of personal information, including her address, telephone number, social security number, current employment specifics, and income history. Claire is provided with an online copy of Astor's privacy policy and is allowed to specify her preferences regarding the disclosure of her personal information. She opts out of any disclosures of her private financial information to companies unaffiliated with Astor, and requests that she not be contacted by telephone or mail with any product or service offers. She does not opt out of email contact from Astor. (See Carnegie's privacy policy, which is applicable to Astor, in Appendix "A.")

### 3.4 CMS Mortgage Broker - Brian

Brian Jones is a mortgage broker with CMS's Los Angeles office looking to develop additional home refinancing business. After the Carnegie-Astor merger and database integration, Brian decides to search the Carnegie database for leads on high-income Astor credit card customers who might be interested in refinancing. Brian executes the following database query to request the contact information for all Astor customers with a Los Angeles County area code and an annual income over \$100,000.

```
SELECT name, address, telephone, ssn, income
FROM customers
WHERE county = 'LA' and income > 100000
```

#### 3.4.1 Results of Query without HDB Active Enforcement

In response to Brian's query, the Carnegie database returns a list of approximately 350 Astor customers in Los Angeles County with an annual income over \$100,000, including Claire. For each customer, Brian obtains an address, telephone number, social security number, and annual income. Using this information, Brian will be able to run credit checks on each of these customers and build a list of potential refinancing leads. Brian asks his staff to contact each one of these customers and give them the CMS sales pitch. Brian can also sell this list to marketing firms that are eager to acquire lists of high-income, credit-worthy individuals. This disclosure of personal information to CMS

does not comport with the customers’ stated privacy preferences.

Name	Address	Telephone	SSN	Email	Income
Andy Andrews	1 Monty St, LA	213-555-5698	589-32-2399	andy@att.net	112,000
.....	.....	.....	.....		.....
.....	.....	.....	.....		.....
Claire Young	605 1 <sup>st</sup> St, SM	310-555-4567	204-23-1234	cy@sbc.net	200,000

*Table A: Brian’s results without HDB Active Enforcement*

### 3.4.2 Results of Query with HDB Active Enforcement

With HDB enforcement in effect, the Carnegie database returns only 120 names in response to Brian’s query. All results returned are consistent with Carnegie’s privacy policy and the opt-out preferences selected by each customer. Because Claire opted out of disclosing her address, telephone number, and social security number, only her name and email address may be revealed to Brian. For customers that accepted Carnegie’s privacy policy and did not opt out of any disclosures, all of their information is revealed. No information is returned for customers that opted out of any marketing communications.

Name	Address	Telephone	SSN	Email	Income
Andy Andrews	1 Monty St, LA	213-555-5698	-	-	112,000
.....	.....	.....	.....		.....
Claire Young	-	-	-	cy@sbc.net	200,000

*Table B: Brian’s results with HDB Active Enforcement*

## 3.5 Compliance Verification Using Eunomia Auditing

### 3.5.1 Claire’s Complaint

Several months after Brian’s query, Claire receives a series of emails from CMS offering to refinance her existing home mortgage. Around the same time, Claire receives many telephone and mail solicitations from a variety of financial service companies not affiliated with Carnegie. Claire suspects that Carnegie has sold her information to outside marketing companies and threatens to initiate legal action against Carnegie for these alleged privacy breaches. The company is unaware of any such disclosures between Astor and CMS, but is concerned about its exposure to liability if such privacy

breaches are a systemic problem. Carnegie decides to initiate an investigation.

### 3.5.2 Investigative Audit

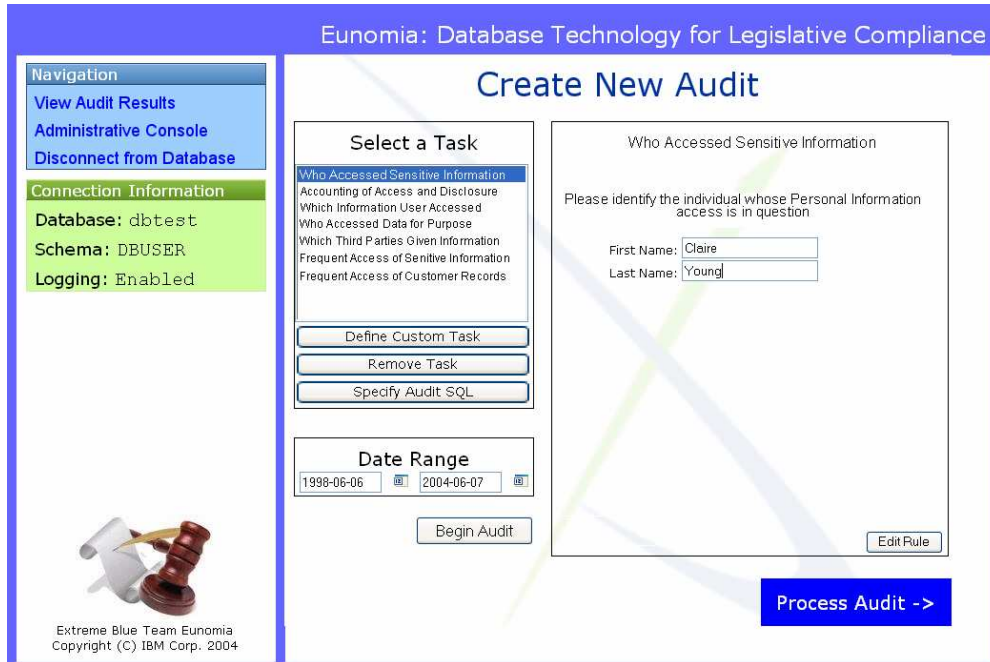


Figure 4: Screenshot of audit application

Using the Eunomia audit application (Figure 4), Carnegie’s Chief Privacy Officer, James Cane, requests an audit of all access to Claire’s records since the date of her Astor credit card application. Eunomia returns an audit trail of all queries that accessed Claire’s records within the specified date range, including user, date, recipient, purpose, and actual information disclosed. The results of the audit are displayed as follows.

User	Date	Customer	Purpose	Recipient	Information
Doug Allen	6-10-03	Claire Young	Cust. Service	None	SSN, telephone ...
.....	.....	.....	.....		.....
Eric Williams	8-03-03	Claire Young	A/R	None	balance, address ...

Table C: All queries accessing Claire’s record

James then narrows the request to reveal only the results of those queries that requested access to Claire’s phone number, mailing address, or email, including aggregate queries. Eunomia provides the following, more specific results.

User	Date	Customer	Phone	Address	Email
Brian Jones	09-10-03	Claire Young	-	-	<a href="mailto:cy@sbc.net">cy@sbc.net</a>
.....	.....	.....	.....		.....
Betty Watson	06-13-03	Claire Young	310-555-5483	38 Montana, SM	<a href="mailto:cy@sbc.net">cy@sbc.net</a>
Betty Watson	07-12-03	Claire Young	310-555-5483	38 Montana, SM	<a href="mailto:cy@sbc.net">cy@sbc.net</a>
Betty Watson	08-17-03	Claire Young	310-555-5483	38 Montana, SM	<a href="mailto:cy@sbc.net">cy@sbc.net</a>

*Table D: Refined results*

James determines that Brian’s query accessed her email address and may have been the source of the email solicitations from CMS. However, the audit trail confirms that Brian’s query did not return Claire’s telephone number or mailing address. The audit trail also reveals several suspicious queries of Claire’s record by Betty, an Astor account manager. It shows that Betty queried the database three times and accessed Claire’s transaction history and contact information. However, the telephone number and mailing address revealed to Betty were for Claire’s former residence. Since the time of Betty’s last query, Claire has changed her contact information in the Astor database. Therefore, it appears that none of the Carnegie companies are responsible for an unauthorized disclosure of Claire’s address and telephone number.

**3.5.3 Audit Report**

After running the audit trail, James prints out an audit report which details all access to Claire’s personal information, including the exact information disclosed in response to each query. At his discretion, James may provide this report to Claire as evidence of Carnegie’s compliance with its privacy policy.

**3.5.4 Post-Audit Investigation**

In this example, the audit confirms that no Carnegie employees accessed Claire’s mailing address and telephone number within the time period at issue. However, the situation would have been different if an employee with legitimate access privileges had reviewed this information during the specified time period. No automated compliance system can determine whether a person with legitimate access privileges has disclosed information outside of the system. For example, Claire’s account representative could have accessed the account information for all of her clients, written it down on paper, and

sold the customer list to a marketing firm.

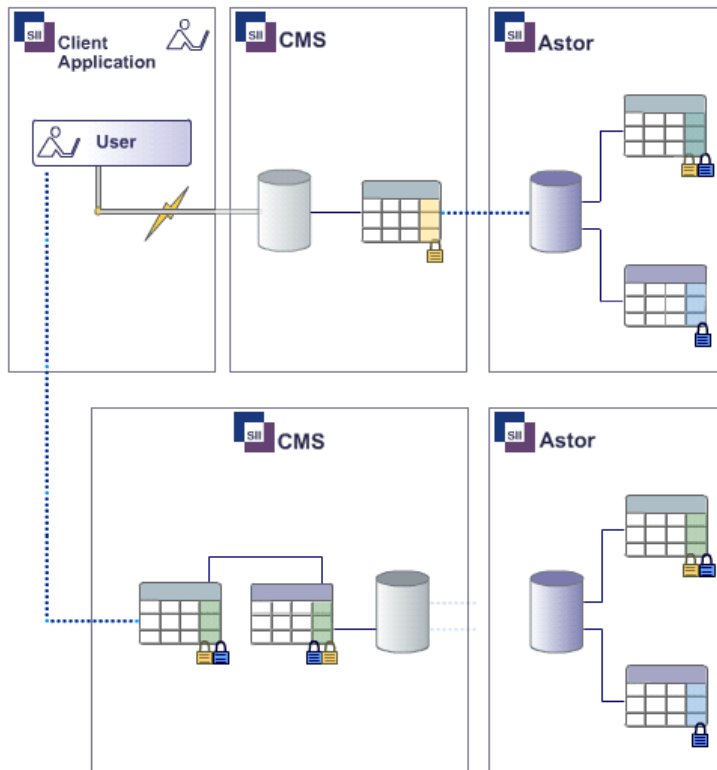
If James had discovered any suspicious access of Claire's current contact information, he could have proceeded by interviewing the suspicious user and/or reviewing his or her email transactions around the time of the query. If there was an inappropriate disclosure, the Eunomia audit application would help Carnegie to identify the source of the leak and take appropriate action to deal with the responsible individual. This situation may also alert Carnegie to a need to adjust its enforcement policies or access privileges. Carnegie could also insert audit flags, prompting Eunomia to alert it to instances of suspicious access, such as queries that request a certain number or type of records. For example, this could prompt Eunomia to notify James if a particular account representative requests contact information for over 50 customers at a time.

### 3.6 Sharing Information with SII

Roger Abrams is a risk analyst for CMS who would like to determine whether there is a correlation between credit card spending patterns and delinquent mortgage payments. This information would help CMS better evaluate the default risk for each prospective home mortgage borrower. Such an analysis would require Roger to join default information from CMS's loan portfolio with credit card customer records from a bank with common customers, based on some personally identifiable information, such as the customer name or social security number. However, banks disclosing customer transaction information violate data disclosure laws and their own privacy policies. Similarly, CMS is also constrained from disclosing personally identifiable customer information to a bank. SII eliminates this problem and allows Roger to conduct this statistical analysis, using CMS and Astor's customer data, without revealing any private information between the two parties.

Roger performs his statistical analysis as follows (Figure 5). Initially, he uses the SII client application to access the CMS application. CMS encrypts its database using its key and commutative encryption function. The encrypted database is then sent to Astor, which encrypts (1) its own database with its key and commutative encryption function, and (2) the database sent to it by CMS with the same key and function. Astor sends both

databases to the CMS application, which encrypts the Astor database and does the necessary operations, in this case a join, using both doubly encrypted databases.



*Figure 5: Sovereign Information Integration between CMS and Astor*

As all sensitive data is encrypted prior to transfer, all the collaborative operations can be performed without fear of a privacy or security breach. Hence, CMS is able to perform valuable statistical analysis that was not possible without SII.

## 4 Technical Discussion

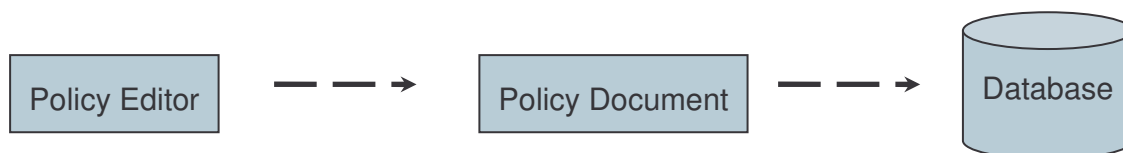
In this section, we elaborate on the technical details of the HDB Active Enforcement, Compliance Auditing, and Sovereign Information Integration, including architectural specifications and performance evaluations.

### 4.1 HDB Active Enforcement

Figure 1 on page 5 provides an overview of HDB's Active Enforcement architecture. In the sections that follow, we describe the three major phases of active enforcement – (1) policy installation, (2) user preferences and data collection, and (3) query enforcement. We also state our assumptions and provide a performance evaluation.

#### 4.1.1 Policy Installation

A company privacy officer begins the policy installation process by creating a data disclosure policy using a simple graphic interface (GUI). The policy editor then creates an arbitrarily complex policy document, written in a privacy language such as P3P or EPAL. The policy translator extracts the relevant information and creates disclosure policy tables that are stored in the database.

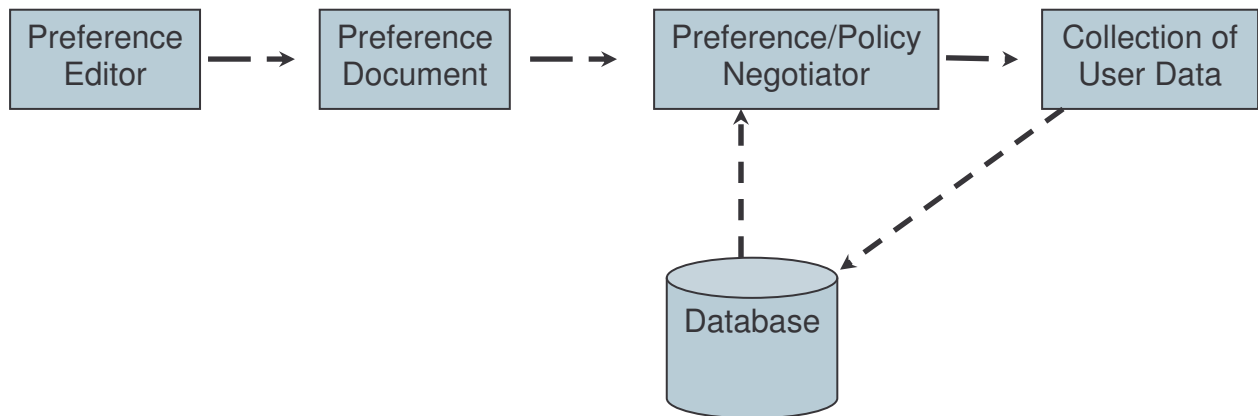


*Figure 6: Policy Installation*

#### 4.1.2 User Preferences and Data Collection

The customer uses a preference creator GUI to specify his user preferences. The preference editor creates a preference document written in XPref, a language that utilizes XPath expressions. A servlet imports the company's policy document from the database and extracts the XPath expressions. It then uses an XPath engine (Xalan) to find matches between the company policy and user preferences. If any conflicts exist, no user data

will be collected. If there are no conflicts, the customer will be asked to input her information in a web interface form, which is submitted to the server. The relevant information is extracted and stored into the database. The following diagram illustrates this preference creation and matching process.



*Figure 7: User Preferences and Data Collection*

#### **4.1.3 Query Enforcement**

To enforce data disclosure policies at the database level, we define an HJDBC driver that encapsulates SQL query parsing, rewriting, and privacy enforcement. The HJDBC driver is wrapped around existing JDBC driver functions, which incorporate disclosure enforcement code into these wrapper functions. The solution is database and application agnostic because the enforcement code resides in the JDBC layer. Accordingly, HDB cell-level enforcement is possible for any company with JDBC-compatible applications and a data source that uses SQL. Another SQL interface can be used in place of JDBC.

#### **4.1.4 Component Assumptions**

We assume that HDB Active Enforcement technology will be deployed in a typical enterprise environment, where the inclusion of changes to policy, opt-in/opt-out choices, user preferences and user data are governed by organizational latency policies. Thus, the act of changing policy, choices, preferences or user data will not lead to

synchronization problems. Also, we assume that applications access the data source through a SQL interface, which is standard in a typical enterprise environment. HDB Active Enforcement will not allow backdoor access (via CLI commands) to the data source when the technology is integrated into the database infrastructure.

#### 4.1.5 Performance

We evaluated the performance of our architecture and enforcement algorithm by using a dataset based on the Wisconsin Benchmark. We conducted experiments on a single 750 MHz Intel Pentium machine with 1 GB of physical memory, setting the buffer pool size to 50MB and the pre-fetch size to 64KB. The goals of our experiments were to measure (1) the overhead of disclosure enforcement and (2) the impact of filtering. To accomplish this, we varied: (1) choice selectivity – the percentage of users who choose to share data for a particular purpose and recipient, and (2) application selectivity – the percentage of records selected by an issued query.

##### 4.1.5.1 Overhead of Disclosure Enforcement

To test the overhead involved in disclosure enforcement, we factored out choice selectivity, varied the application selectivity, and observed the execution times of both the original and rewritten queries. We set the choice selectivity to 1.0, which is the *worst case scenario*. In this scenario, we incur the cost of disclosure processing, but do not see any performance gains from filtering prohibited records from the result set.

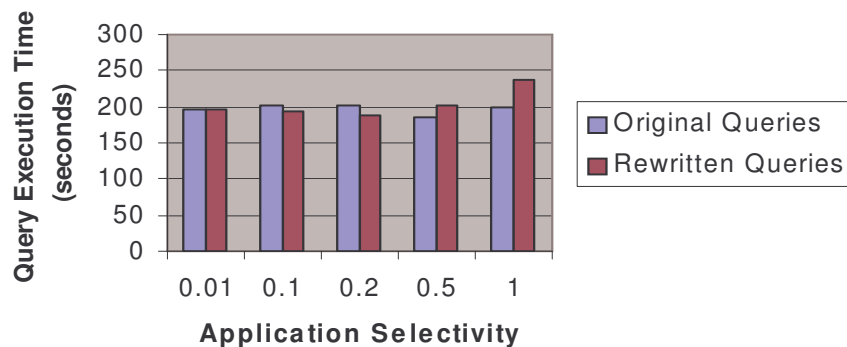


Table E: Performance Overhead in Worse-Case Scenario

Table E above shows the overhead cost of performing disclosure enforcement in the worst case scenario on a table containing 10 million records. As it indicates, the overhead introduced in this case is minimal. When all the records are selected, a 15% slowdown is experienced. In the average case, the rewritten query executes within similar timescales to the original query.

#### 4.1.5.2 Impact of Filtering

In cases with choice selectivity less than 100% (which should be the typical enterprise scenario), the rewritten queries perform significantly better than the original queries because the rewritten queries need to read fewer records. The table below shows the results of our experiments when we set the application selectivity to 1.0 and varied the choice selectivity.

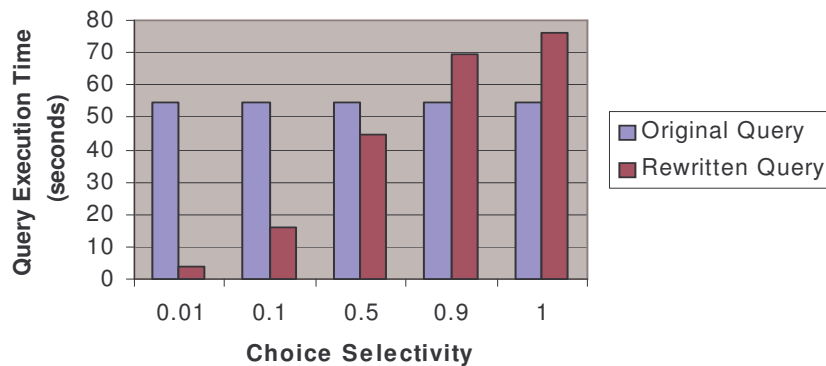


Table F: Cost Comparison of Original vs. Rewritten Queries

When the choice selectivity is low, the rewritten query executes 10+ times faster than the original query. As we increase the choice selectivity, we do derive gains from filtering. However, when the choice selectivity is near 100% (i.e. near 1.0), we incur the cost of disclosure enforcement, but do not benefit from choice selectivity. We have performed other experiments at application selectivity levels 0.01, 0.10 and 0.50 that further support these results.

Thus, when choice selectivity is low and application selectivity is high (the *best case* scenario) the HDB generated query executed 10+ times faster than the original

query. In the *worst case*, there is a slowdown of 5 -15%. On average, the rewritten and original queries have similar execution timescales and performance characteristics.

## 4.2 HDB Eunomia Compliance Auditing

Eunomia Compliance Auditing consists of two components: (1) a logical logging system that records all queries and changes to the database; and (2) an audit application that reconstructs the state of the database at any given time and provides detailed audit trails of data access and disclosure in response to a specified audit expression.

### 4.2.1 Logical Logging

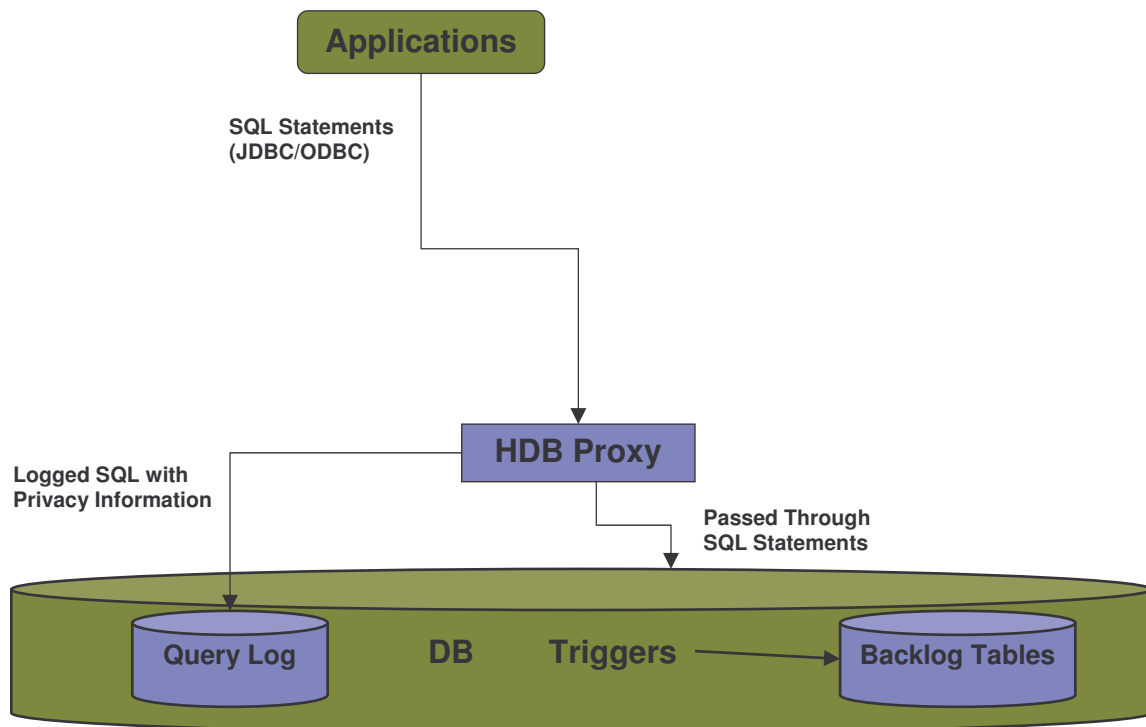


Figure 8: HDB Logical Logging

HDB’s “logical logging” system captures all transactions with the database. From the enterprise applications, the HDB proxy captures and records the query executed, time of execution, user identification, purpose of the request, and external third party data recipients (if any). The purpose of the request can be inferred from the application if it is

not specified. Backlog tables, which are maintained by database triggers, record all insertions, deletions, and updates to the database. A logging setup tool reads table schemas and automatically generates the query logs and backlog tables. The data stored in the query logs and backlog tables allow exact database reconstruction at any point in time, up to the desired retention period. Logical logging provides all necessary information to precisely determine which queries, if any, disclosed specified data. At audit time, auditor can specify the precise data to be audited.

#### 4.2.2 HDB Audit Application

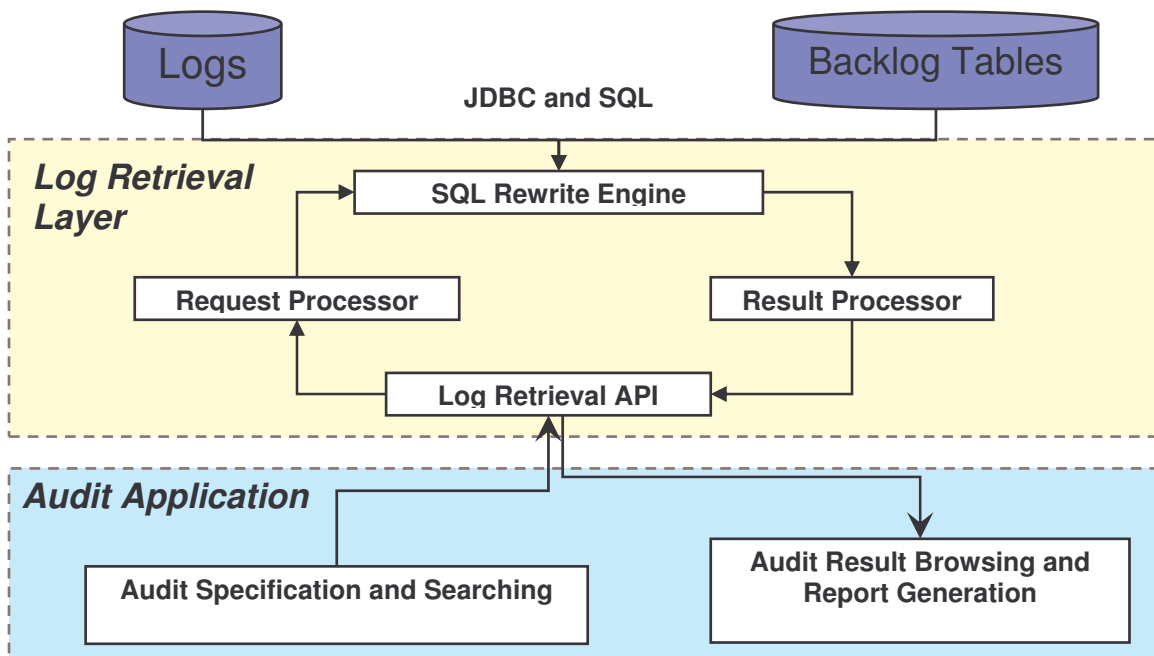


Figure 9: HDB Audit Application

The HDB audit application supports a variety of search parameters, allowing the auditor to declaratively specify, through SQL statements in an audit expression, the data to be audited. It uses backlogged records to reconstruct the state of the database at the time a particular query was issued. In response to an audit query, the log retrieval layer selects the records from the backlog with a timestamp earlier than the timestamp of the query. From that set, it selects the records with the most recent timestamp. It then selects from this set only the records that may have contained the data. The log retrieval API reruns the query formed using conditions from original query on this data. The HDB

audit application uses a SQL rewrite engine, which combines an audit expression with a logged query onto a new query with a result that is non-empty if and only if that query disclosed some data in the audit expression. For each query found to disclose specific data, HDB produces an audit trail specifying the user, purpose, recipient, query date, and exact (cell level) information disclosed.

### 4.2.3 Component Assumptions

Auditing compliance for standard disclosure policies (e.g. security or privacy policies) requires only audit information and algorithms for analyzing this information. However, for legislative compliance auditing, the requirements are more nebulous due to the nature of legislation. Thus, a solution that seeks to address all this problem area in a uniform way must be based on a generic, extensible audit model. HDB's Compliance Auditing component takes into account the vagueness of legislation and assumes that the core directives can be specified in a XML-like specification.

- **Schema Changes:** The current prototype of HDB Compliance Auditing assumes there are no schema changes, although we recognize that the frequency and complexity of schema changes in certain environments may increase the difficulty of reconstructing past database states. This is a non-trivial, but solvable issue that can be addressed in conjunction with the customer during implementation.
- **Alternatives to Triggers:** The proposed HDB Compliance Auditing architecture assumes that triggers will be used to populate backlog tables and query logs, which are stored in the database, although we acknowledge that certain database and legacy systems may not support the use of triggers. Accordingly, we have also proposed an alternative architecture in which database recovery logs (such as those maintained by legacy database systems), instead of triggers, are used for this purpose. There are also other options.

### 4.2.4 Performance

Our performance experiments are based on the auditing architecture presented in section 4.2.1 (i.e. using database triggers). The goals of our experiments were to (1) determine the cost of including the backlog tables on inserts, updates and deletes, and (2)

quantify the penalty incurred by having a backlog when checking whether a query disclosed specified data. For our experiments, we used the TPCB benchmark.

#### 4.2.4.1 Penalty on Inserts, Updates, Deletes

Using the supplier table of TPCB and executing 10,000 insert, delete and update operations, with and without backlogging, we obtained the following results:

	Insert	Update	Delete
Original	57	33	30
Backlog (no index)	150	161	130
Backlog (index1)	152	161	130
Backlog (index2)	152	161	130

*Table G: Penalty on Inserts, Updates and Deletes (in Seconds)*

In the above table, we see that an insert operation is slowed down by a factor of 3, an update is slowed down by a factor of 5 and a delete is slowed down by a factor of 4. Indexing the backlog on various attributes has little effect on the slowdown. These slowdown factors are significantly better than those introduced in result logging systems.

#### 4.2.4.2 Penalty on Checking for Disclosure

Again, we used the supplier table and tested a query of form: *select \* from tpch.supplier where s supkey = l*. The following were our results:

	Time
Original	3
Backlog (no index)	58
Backlog (index1)	5
Backlog (index2)	5

*Table H: Penalty on Including Backlog Tables for Checking*

A query with a similar form to a standard audit query is executed without the backlog table and then with the backlog table. Without indexing the backlogged version runs 20 times slower than the original. When indexed, the backlogged version runs on within timescales comparable to the original version.

### 4.3 HDB Sovereign Information Integration

As mentioned above, the goal of the information sharing component of HDB technology is to allow for the computation of query results across autonomous data sources without revealing any information other than the results.

#### 4.3.1 Architecture

Figure 3 on page 9 depicts the basic architecture of the SII platform, which is comprised of the following components:

- **SII Server** - provides the necessary functionality on the data provider side to enable secure information sharing. This includes components for query processing (cryptographic protocols for basic operations), access verification, data encryption, and communication with other data providers. It also binds applications to purposes for a purpose-based access control mechanism. The server also maintains metadata needed for processing a query issued by an application, including viewing information to retrieve data from the data provider database, database access information, and context information.
- **SII Client** - provides the necessary functionality to map a client's schema to a data provider's schema, construct and invoke web service query requests against multiple data providers, and receive web service responses. The SII client uses the client metadata database to store and retrieve mapping information and data provider access information.
- **SII Application** – is a thin layer on top of the SII client, which invokes the required SII operations. For example, an application providing epidemiological results for drug reactions may invoke intersection size operations multiple times applying different filter predicates. The application also provides a communication interface to a SII user and interacts with the SII client using the SII client API.

#### 4.3.2 Performance

Encryption time dominates the performance of the SII operations. However, other components may also become computationally expensive if not implemented carefully. For instance, we discovered that parsing database tables into XML web service

arguments using Axis SOAP engine requires more time than encryption. Therefore, we attach tables as binary files to avoid parsing the tables. The encryption algorithm we use requires modular exponentiation. We used AEP Runner 2000 SSL cards to speed up exponentiation.

Encryption Engine	Number of rows in a table		
	1,000	5,000	10,000
CPU Intel III 2.0 GHz	34s	175s	320s
AEP Runner 2000	3.5s	19s	37s

*Table 1: Increase in Speed Gained by Using SSL Card*

The table above shows the performance acceleration attained by using the AEP Runner cards. Our implementation employs a User Defined Function (UDF) to perform encryption. A standard (scalar) UDF, as a program, has a scope over only one row from an input table at a time. However, AEP Runner can gain the advertised speeds only if multiple (100-200) threads simultaneously post exponentiation requests to the card's API. Therefore, we used a table UDF that can operate on several input rows at a time. Our UDF fetches  $k$  rows at a time, and creates a separate thread for every row. Once all threads have finished, the encrypted rows are returned to the database and next  $k$  rows are fetched.

Our experiments show that AEP Runner can speed up the execution of the UDF performing encryption by an order of magnitude. In our performance experiments of different real world scenarios (e.g. information integration with Homeland Security) we were able to reduce the execution time by a factor for each accelerator card added. For example, an operation between two organizations with 10 million records each would take around 120 minutes with one accelerator card; 12 minutes with ten cards; and 1.2 minutes with twenty cards.

An additional feature of the AEP Runner API is that an application does not have to know how many cards are installed in the computer. The AEP scheduler distributes exponentiation requests between the cards automatically. Thus, we can expect a linear increase in speed with an increasing number of cards.

## 5 Conclusion

Hippocratic Database technologies offer efficient methods of managing, auditing, and transmitting electronic data in a manner that preserves the privacy of individual customers. HDB provides: (1) active enforcement of negotiated privacy policies at the database level; (2) efficient data access tracking that identifies all who have accessed each cell of the database and any modifications made; and (3) information sharing across autonomous data sources that provides the results of the query without revealing any other data. We trust that these HDB concepts will serve as a model for future data management and exchange solutions in the financial services industry.

## Appendix A: Privacy Policy for Carnegie Astor & Co.

This Privacy Policy covers the entire Carnegie Astor & Co. (“Carnegie”) family of companies, including Carnegie Investment Bank, Carnegie Mortgage Services, JJ Astor Bank USA, and JJ Astor Insurance Company.

### Information Collected

Carnegie collects information from a variety of sources to help us provide the highest quality services to you and best meet your personal needs.

- We collect information from your requests for Carnegie products and services, such as credit card or loan applications.
- We collect information from your transactions with Carnegie as well as other companies.
- We collect information about your credit history from credit bureaus and similar services.

### Security of Information

Carnegie takes several measures to protect your privacy and secure your confidential information. Here are some examples.

- We keep information under physical, electronic and procedural controls that comply with federal standards. These controls help keep to information from being changed or destroyed.
- We prohibit our employees from using information about you for anything other than Carnegie business.
- We require companies working for us to protect information. They agree to use it only to provide the services we ask them to perform for you and for us.

### Sharing of Information

We may share information about you within the Carnegie family of companies. This helps us to offer you financial products and services such as loans, deposits, investments and insurance.

We may share information about you with outside companies that work for Carnegie, which may include marketing firms. We may also share information about you with outside financial service companies that have joint marketing agreements with us. These agreements permit you to get additional product and service offers.

However, we strictly limit the information we share with companies making non-financial offers.

- We may share only your name, address and phone number with companies for these offers.
- We may share information about your Carnegie auto loan or lease with your auto dealer and auto maker for auto offers.
- We may share information about you with our co-brand "partners" so that they can market non-financial products to you.

We may also share information about you as required or permitted by law. This allows us to share for legal and routine business reasons. Here are some examples.

- We may share information with regulators and law enforcement officials.
- We may share information to protect you, Carnegie and others against fraud.
- We may share account activity with credit bureaus.
- We may share information with your consent.
- We may share information such as account name and number with check printers and with others that provide services to you or to us.

### Choices Regarding Information Sharing

We offer you the following choices about sharing information that identifies you.

- **Choice One.** You may tell us not to share within the Carnegie family of companies: (i) information from you or from others for determining your eligibility for products, or (ii) information from credit bureau reports for marketing purposes. Even if you make this choice, we may continue other information sharing within Carnegie. For example, in the course of business, we may continue to share name and address, information about transactions with us, as well as survey or similar information.
- **Choice Two.** You may tell us not to share information about you for non-financial offers described above. If you do and are in any of our co-brand programs, we may continue to share information about you with co-brand “partners” to provide the program to you.

### Making Your Privacy Choices

You may contact us to provide your privacy choices as stated in the Carnegie Privacy Policy. Also, you may tell us not to contact you with telephone or mail offers or you may change prior choices. **To do so, submit your privacy choices online or contact us at 1-800-555-1212.** Please have your account number ready.

If you make any of the choices listed, you may still get offers with your account statements and when you contact us. You may also get offers in connection with our maintaining and servicing your account relationship.

### Privacy Choices

1. Please do not share within the Carnegie family of companies:
  - Information from me or from others for determining my eligibility for products, or
  - Information from credit bureau reports for marketing purposes.
2. Please do not share information about me with companies for offers of non-financial products and services.
3. Please do not contact me with offers of products or services by mail.
4. Please do not contact me with offers of products or services by telephone.
5. Please do not contact me with offers of products and services by email.

## References

- 
- [1] R. Agrawal, J. Kiernan, R. Srikant and Y. Xu. "Hippocratic Databases". *Proc. of the 28th Int'l Conf. on Very Large Databases*, Hong Kong, China, August 2002.
- [2] K. Lefevre, R. Agrawal, V. Ercegovac, R. Ramakrishnan, Y. Xu, D. DeWitt. "Limiting Disclosure in Hippocratic Databases". *Proc. of the 30th Int'l Conf. on Very Large Databases*, Toronto, Canada, August 2004.
- [3] R. Agrawal, P. Bird, T. Grandison, J. Kiernan, S. Logan, W. Rjaibi, "Extending Relational Database Systems to Automatically Enforce Privacy Policies". *Proc. of the 21<sup>st</sup> Annual Conf. on Data Engineering*, Tokyo, Japan, April 2005.
- [4] R. Agrawal, R. Bayardo, C. Faloutsos, J. Kiernan, R. Rantzau and R. Srikant. "Auditing Compliance with a Hippocratic Database". *Proc. of the 30th Int'l Conf. on Very Large Databases*, Toronto, Canada, August 2004.
- [5] R. Agrawal, A. Evfimievski and R. Srikant. "Information Sharing across Private Databases". *Proc. of the ACM SIGMOD Conference on Management of Data*, San Diego, California, June 2003.
- [6] R. Agrawal, D. Asonov, P. Baliga, L. Liang, B. Porst, R. Srikant. "A Reusable Platform for Building Sovereign Information Sharing Applications". *Proc. of the 1<sup>st</sup> Workshop on Databases in Virtual Organizations*, Paris, France, June 2004.