Hippocratic PostgreSQL*

Jalaja Padma[1], Yasin N. Silva[1], Muhammad U. Arshad[2], Walid G. Aref[1]

[1]*Department of Computer Science*
Purdue University, West Lafayette, IN, USA
{jpadma, ysilva, aref}@cs.purdue.edu

[2]*Department of Electrical & Computer Engineering*
Purdue University, West Lafayette, IN, USA
{marshad}@ecn.purdue.edu

***Abstract*—Privacy preservation has become an important requirement in information systems that deal with personal data. In many cases this requirement is imposed by laws that recognize the right of data owners to control whom their information is shared with and the purposes for which it can be shared. Hippocratic databases have been proposed as an answer to this privacy requirement; they extend the architecture of standard DBMSs with components that ensure personal data is handled in compliance with its associated privacy definitions. Previous work in Hippocratic databases has proposed the design of some of these components. Unfortunately, there has not been much work done to implement these components as an integral part of a DBMS and study the problems faced to realize the Hippocratic databases. The main goal of the 'Hippocratic PostgreSQL' project is to perform this implementation and study. The project includes the implementation of components to support limited disclosure, limited retention time, and management of multiple policies and policy versions. This demo presents the use of these components both from a terminal-based SQL command interface and through a Web-based healthcare application that makes use of the implemented database-level privacy features. Hippocratic PostgreSQL has the novel feature of augmenting both *k*-anonymity and generalization hierarchies into the Hippocratic DBMS engine functionality. Several interesting problems emerge as a result and their solutions are presented in the context of this demo.**

I. INTRODUCTION

Privacy has become an important component in systems that handle personal data. Several research efforts have studied ways to make data management systems aware of privacy requirements, e.g., Hippocratic databases, anonymization and generalization, and privacy-preserving data mining.

Agrawal et al. proposed the Hippocratic database model to address the privacy requirements inside of DBMSs [1]. One of the main benefits of the Hippocratic database model is that privacy requirements are fulfilled at the database-level. Companies and organizations that use a Hippocratic database can make use of these privacy-preserving features instead of implementing ad-hoc application-level systems to comply with privacy laws, e.g., the Health Insurance Portability and Accountability Act (HIPAA) in the U.S.A., or the Access to Health Records Act in the United Kingdom. Advantages of the implementation of authorization mechanisms at the database-level are further discussed in [7].

A general design has been proposed for some of the key components of the Hippocratic database [1]-[4]. The main Hippocratic database guidelines are introduced in [1] while the work in [2] presents an initial design for the limiting disclosure. Research has been conducted to study the problems in realizing Hippocratic databases, e.g., [4], [6]. Previous implementations have the form of middleware systems between the application and database levels rather than integral components of a DBMS. A demonstration of some Hippocratic database features is presented in [6], which focuses only on the support of limited disclosure on the `SELECT` operation, support of a single policy, and limited data collection. Chaudhuri et al. propose extending the SQL Grant command to allow fine-grained authorization [7]. Fine-grained authorization is a step towards enforcing privacy-based limited disclosure at the database-level. However, fine-grained authorization needs to be extended to support privacy policies as a unit, which may contain multiple fine-grained authorization commands. In contrast, a Hippocratic database operates on an input privacy policy and treats it as a unit, e.g., translate, enable, disable, or comply with it. Anonymization is another area of active research related to privacy. Several algorithms have been proposed to achieve *k*-anonymity. To the best of our knowledge, none of these algorithms have been implemented at the database-level. We study the functioning of the same at the database-level and propose and implement an algorithm for it within the PostgreSQL engine. The generalization of sensitive attributes based on personal preferences, and its implementation outside of the database engine are proposed in [9].

The main contributions of our work are (i) the design of several Hippocratic database privacy components – policy management, limited disclosure, limited retention time that integrate and improve previous work, (ii) the design of generalization and *k*-anonymity components that work together with limited disclosure and their implementation as integral sub-systems of a DBMS, (iii) the study of realization-related problems faced during this implementation. The first stage of the Hippocratic PostgreSQL project focuses on the implementation of components to support limited disclosure on multiple DML operations, sensitive attribute generalization, *k*-anonymity, limited retention time, and support for multiple policies and policy versions.

* This work is partially supported by NSF Grant Number IIS-0811954.

The demonstration of Hippocratic PostgreSQL shows the use of the new and extended SQL commands and privacy facilities from both a terminal-based interface and a Web-based healthcare application. The healthcare application allows doctors and nurses to handle medical electronic records and allows the patients to specify their preferences. The Hippocratic PostgreSQL can support multiple applications from different domains. The only effort to be put in would be the creation of appropriate data tables and new policies.

II. HIPPOCRATIC POSTGRESQL ARCHITECTURE

Table 4 in Fig. 2 presents the schema and partial content of table `Patient` used in our examples. This table stores one row per data owner. The primary key of this table identifies uniquely a data owner in the whole database. We call this table the primary table. The architecture of Hippocratic PostgreSQL is given in Fig. 1.

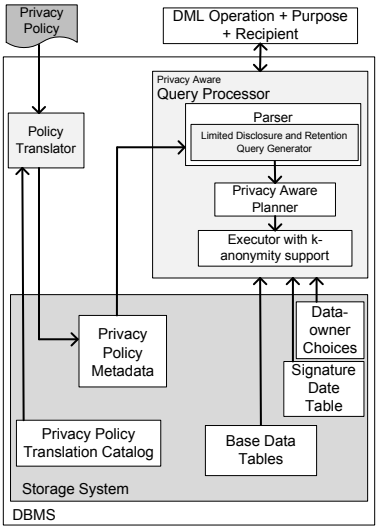

Fig. 1 Hippocratic PostgreSQL Architecture

A. Privacy Policy Management

The '**Policy Translator'** module translates an electronic version of the privacy policy into its database equivalent. In general, a policy can be specified using any privacy policy specification language. The current version of Hippocratic PostgreSQL supports P3P [5], a standard language proposed by the W3C. The following Hippocratic PostgreSQL command is used to perform the translation:

```
TRANSLATE POLICY <policy-path> [FROM
<language>] [POLICY_ID <policy-id>
POLICY_VERSION <policy-version>]
```

where <policy-path> is the path of the privacy policy in the underlying file system. The default value of <language> is P3P. Additionally, the policy id and policy version can be specified directly. Otherwise, the translator will assign a policy id and a version number automatically.

Given that a privacy policy is expressed in terms of data elements and retention time periods using the terminology of a specific privacy language, the translation process requires information to map this terminology to tables, columns, and specific time periods in the database. The mapping information is stored in the '**Privacy Policy Translation Catalog'** (Fig. 2, Table1). The result of the translation process, i.e., the database version of the policy, is stored in the '**Privacy Policy Metadata'** tables (Fig. 2, Table 2-3).

Several policies can be active simultaneously as long as each policy has a different primary table, e.g., `Doctors`, `Contractors`, etc. Furthermore, Hippocratic PostgreSQL supports multiple versions of the same policy simultaneously. This feature requires extending the primary table with an additional attribute that specifies the current policy associated with each data owner. The support for multiple policies and policy versions is a key feature for real world systems. As stated in [4], 80% of the organizations use a different privacy policy for employees and clients while 75% require support of policy versions. Information about the current policies translated into Hippocratic PostgreSQL and their status can be obtained by querying the system table `PG_PrivacyPolicy`.

B. Integrated Limited Disclosure

Hippocratic PostgreSQL's '**Privacy-Aware Query Processor'** is an extension of the regular query engine which ensures that any database access complies with the active privacy policies stored in the **'Privacy Policy Metadata'** tables and the data owner preferences. The design and implementation of the Privacy Aware Query Processor integrates and extends the ideas in [1], [2], [4]. All the DML operations are extended to allow the specification of their associated purposes and recipients in the following way

```
<DML-Operation> PURPOSE <Purpose>
RECIPIENT <Recipient>
```

The following `SELECT` operation obtains information about patients and their diseases that will be shared with a research lab.

```
SELECT P.name, P.sex, P.disease FROM
PATIENT P PURPOSE research RECIPIENT lab
```

The result of the previous `SELECT` operation will be restricted to include only the columns that the combination of purpose and recipient is allowed to access according to the policy specification. It will be further restricted to include only data of patients who opted-in to share their disease information with research labs, if such an opt-in choice is allowed by the policy. Intermediate Result 1 in Fig 2 is an example depicting limited disclosure.

For other DML operations, i.e., `INSERT`, `UPDATE`, and `DELETE`, the recipient represents the entity that triggers the operation. The execution of an `INSERT`, `UPDATE` or `DELETE` operation can be allowed, denied or allowed with limited effect by the Privacy-Aware Query Processor. In this case, the effect of the operation is restricted to the subset of the data to which a user has access to, i.e., allowed by the policy, data owner preferences and retention time specifications. An `INSERT` operation on Table `T` requires the user executing the operation to have access to all the columns of `T` except the ones that receive the value `NULL`.

PRIVACY MAPPINGS				
purpose	**recipient**	**P3P type**	**table_name**	**col_name**
Treatment	Doctors	#patient.pno	options_patient	pno_option

Table 1: Stores the mappings between P3P element and database table.

POLICIES		
policy_id	**primary table**	**id_column_name**
Medical PatientPolicy1.0.0.1	PATIENT	pno

Table 2: Stores the policies (could be active/inactive)

PATIENT_CHOICES					
pno	**pno_ option**	**name_ option**	**birthsex_ option**	**SA Gen-Level**	**phone_ option**
P1	t	t	t	1	t
P2	t	f	t	1	t
P3	t	t	f	0	f
P4	t	f	f	0	f

Table 3: Stores the opt-in/out choices of each data owner. (One/more tables specific to each application

PATIENT					
pno	**name**	**birth**	**sex**	**phone**	**disease**
P1	N1	1965	M	765 111 1111	Gastritis
P2	N2	1966	M	765 222 2222	Bronchitis
P3	N3	1967	F	765 333 3333	Stomach Ulcer
P4	N4	1966	F	765 444 4444	Indigestion

Table 4: Primary table that stores the data owner's data.

Example Scenario:

(a) The above tables are created and populated. Domain Generalization Hierarchy for the Sensitive Attribute 'Disease' is stored in another table.

(b) The P3P policy gets translated into the metadata tables POLICIES, PATIENT_CHOICES and a few more not given above.

(c) **Query:** `SELECT pno, phone, disease from patient;`

pno	phone	disease
P1	765 111 1111	Gastritis
P2	765 222 2222	Bronchitis
P3		Stomach Ulcer
P4		Indigestion

Intermediate Result 1: Opt-in/out choices specified in the policy are ensured through limited disclosure component.

pno	phone	disease
P1	*765 xxx xxxx*	Gastritis
P2	*765 xxx xxxx*	Bronchitis
P3		Stomach Ulcer
P4		Indigestion

Intermediate Result 2: After 2-anonymization of Intermediate Result 1.

pno	phone	disease
P1	*765 xxx xxx*	*Stomach Infection*
P2	*765 xxx xxx*	*Respiratory Infection*
P3		Stomach Ulcer
P4		Indigestion

Final Result: After Sensitive Attribute Generalization of Intermediate Result 2.

Fig. 2 Hippocratic PostgreSQL Metadata Table and Healthcare Application Data Tables

Alternatively, if the keyword `STRICT` is specified after `INSERT`, the user executing the operation is required to have access to all the columns of `T`. An `UPDATE` operation on Table `T` requires the user to have access to all columns of `T` being updated. A `DELETE` operation on Table `T` requires the user to have access to all columns of `T`.

C. Integrated Limited Retention

As per the definition for 'Limited Retention' stated in [1], data should be retained only as long as necessary for the fulfillment of the purposes for which it was collected. Even though the deletion of all data items that have outlived their purpose would help comply with *Limited Retention*, completely forgetting information in a database without affecting recovery is non-trivial. Our approach to support retention time is through the privacy policy retention specification. The period for which the information should be accessible is specified through the *Retention* element in the P3P policy. This element can have several predefined values: no-retention, stated-purpose, legal-requirement, constant value etc., The mapping between P3P retention value (*Rt*), purpose and actual time period is stored in the privacy catalog table *PRIVMAP_RETENTION*.

```
SELECT P.name, P.birth, P.sex, P.disease
FROM PATIENT P
```

The above query gets translated internally into the following query to comply with the specified retention period (90 days in this example):

```
SELECT P.name, P.birth, P.sex, P.disease
FROM PATIENT P WHERE current_date <=
((SELECT signature_date from
PATIENT_SIGNATURE_DATE WHERE
PATIENT_SIGNATURE_DATE.pno=P.pno) + 90)
```

The query translator builds a condition that ensures that the date in which a command is executed falls within *Rt* days of the date when the policy was signed.

III. Integrated Sensitive Attribute Generalization and K-anonymity

Privacy through *k*-anonymization ensures that the data owner remains anonymous among at least *k* persons. The idea of Sensitive Attribute Generalization [9] adds the notion of personalized privacy to the concept of *k*-anonymity. We propose and implement both *k*-anonymity and sensitive attribute generalization at the database level. The anonymization algorithm implemented in our system is inspired by the Datafly algorithm [10]. We assume that the data holder has the knowledge of the quasi-identifiers [8] in the data set. The algorithm works by generalizing the quasi-identifiers in the tuples and merging the tuples with the same combination of quasi-identifier values into QI groups; the

process is repeated until every QI group has a frequency $\geq k$. In our algorithm, each user can specify his/her own k value and the generalization level. The generic steps of the anonymization query node are presented in Fig 3. The output of the anonymization node (Fig. 2, Intermediate Result 2) is passed on to the Sensitive Attribute (SA) Generalization Node that modifies the value of the sensitive attribute according to the privacy preferences of the data owner (Fig. 2, Table3). The final output complies with opt-in/opt-out preferences, sensitive attribute generalization preferences and is k-anonymous.

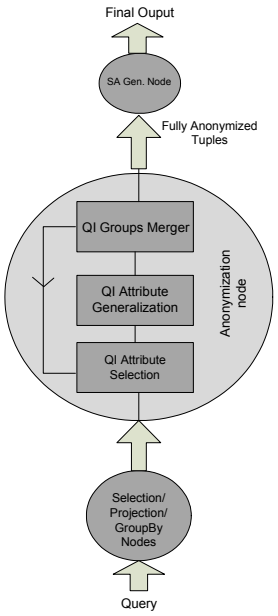

Fig. 3 Query Execution flow with k-anonymity and SA Generalization

IV. Hippocratic PostgreSQL in a Healthcare Application

The demonstration of Hippocratic PostgreSQL includes the presentation of all the commands described in Section II using PostgreSQL's psql terminal-based interface. Additionally, the demonstration presents a Web-based healthcare application that shows how the Hippocratic PostgreSQL privacy related features can be used in practice.

The Web-based healthcare application has two main components. The first one is a system designed to be used by doctors and nurses to manage electronic medical records. This application manages basic information of patients' as-well-as clinic and medication data. The privacy policy allows the doctors of a patient to access all his medical information. Nurses have limited access to certain data elements like home or email addresses. Furthermore, the initial policy requires users to agree to share clinical information for research purposes performed in the medical centre labs. Fig. 4 shows a screen shot of this application where some data items are not released to the nurse reviewing the medical record. To demonstrate the simultaneous use of multiple policy versions, we simulate a change in the privacy law that requires the policy to allow new patients to choose whether or not they want to share their clinic information for research purposes. The report of patients and diseases generated for a research lab releases clinic data of (i) all patients who signed the first version of the policy, and (ii) patients who opted-in to share this information and signed the second version of the policy.

The second component of the presented healthcare application is a system designed to be used by patients. This application gives access to multiple services. Specifically it allows patients to specify or update their personal preferences for the opt-in/opt-out choices allowed by the policy they signed.

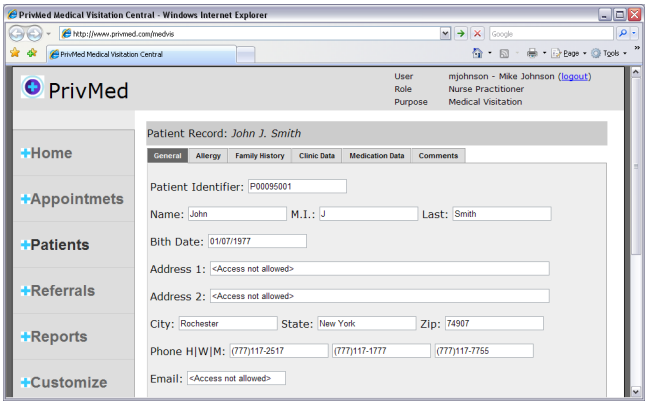

Fig. 4 Use of Hippocratic PostgreSQL at Application Level

V. Future Work

The second stage of the Hippocratic PostgreSQL: Stage-II project will focus on the design and implementation of other important components like: (i) support for openness, (ii) limited collection, (iii) safety, (iv) compliance and (v) limited use.

References

[1] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Hippocratic databases," In *VLDB*, 2002.

[2] K. LeFevre, R. Agrawal, V. Ercegovac, R. Ramakrishnan and Y. Xu, "Limiting disclosure in Hippocratic databases," In *VLDB*, 2004.

[3] R. Agrawal, P. Bird, T. Grandison, J. Kiernan, S. Logan, and W. Rjaib, "Extending relational database systems to automatically enforce privacy policies," In *ICDE*, 2005.

[4] Y. Laura-Silva and Walid G. Aref, "Realizing privacy-preserving features in Hippocratic databases," In *Third International Workshop on Privacy Data Management*, *ICDE*, 2007.

[5] L. Cranor, M. Langheinrich, M. Marchiori, M. Pressler-Marshall, and J. Reagle, "The platform for privacy preferences 1.0 (P3P1.0) specification," *W3C Recommendation*, 2002.

[6] R. Agrawal, A. Kini, K. LeFevre, A. Wang, Y. Xu, and D. Zhou. "Managing healthcare data hippocratically," In *SIGMOD*, 2004. (Demo paper)

[7] S. Chaudhuri, T. Dutta, and S. Sudarshan, "Fine grained authorization through predicated grants," In *ICDE*, 2007.

[8] L. Sweeney, "k-Anonymity: A model for protecting privacy," *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5): 557-570, 2002.

[9] X. Xiao and Y.Tao, "Personalized Privacy Preservation," In *SIGMOD*, 2006.

[10] L. Sweeney, "Guaranteeing anonymity when sharing medical data, the datafly system," *Journal of the American Medical Informatics Association*, 1997.