The VLDB Journal (2008) 17:603–619
DOI 10.1007/s00778-006-0023-0

REGULAR PAPER

# Purpose based access control for privacy protection in relational database systems

**Ji-Won Byun · Ninghui Li**

Received: 30 September 2005 / Accepted: 24 May 2006 / Published online: 1 September 2006


**Abstract** In this article, we present a comprehensive approach for privacy preserving access control based on the notion of purpose. In our model, purpose information associated with a given data element specifies the intended use of the data element. A key feature of our model is that it allows multiple purposes to be associated with each data element and also supports explicit prohibitions, thus allowing privacy officers to specify that some data should not be used for certain purposes. An important issue addressed in this article is the granularity of data labeling, i.e., the units of data with which purposes can be associated. We address this issue in the context of relational databases and propose four different labeling schemes, each providing a different granularity. We also propose an approach to represent purpose information, which results in low storage overhead, and we exploit query modification techniques to support access control based on purpose information. Another contribution of our work is that we address the problem of how to determine the purpose for which certain data are accessed by a given user. Our proposed solution relies on role-based access control (RBAC) models as well as the notion of conditional role which is based on the notions of role attribute and system attribute.

**Keywords** Privacy · Access control · Purpose · Private data management

J.-W. Byun (✉) · N. Li
CERIAS and Department of Computer Science,
Purdue University, West Lafayette, IN, USA
e-mail: byunj@cs.purdue.edu

N. Li
e-mail: ninghui@cs.purdue.edu

## 1 Introduction

While current information technology enables people to carry out their business virtually at any time in any place, it also provides the capability to store various types of information the users reveal during their activities. Indeed, a study conducted by the Federal Trade Commission in May 2000 [12] shows that 97% of web sites were collecting at least one type of identifying information such as name, e-mail address or postal address of consumers. The fact that the personal information can be collected, stored and used without any consent or awareness creates fear for privacy violation for many people.

The advance of database technology has also significantly increased privacy concerns. The current database technology makes it possible to collect and store a massive amount of person-specific data, and the use of innovative knowledge extraction techniques combined with advanced data integration and correlation techniques [10,13,28] makes it possible to automatically extract a large body of information from the available databases and from a large variety of information repositories available on the web.

Even though the direct victims of privacy violations are consumers, many enterprises and organizations are deeply concerned about privacy issues as well. Many companies, such as IBM and the Royal Bank Financial Group, use privacy as a brand differentiator [3]. By demonstrating good privacy practices, many businesses try to build solid trust to customers, thereby attracting more customers [3]. Potential lawsuits by consumers and recently enacted privacy legislations also require organizations to pay close attention to the management of private data.

As privacy becomes a major concern for both consumers and enterprises, many privacy protecting access control models have been proposed [1,3,17,22]. We emphasize that

privacy protection cannot be easily achieved by traditional access control models. The first reason is that while traditional access control models focus on which user is performing which action on which data object, privacy policies are concerned with which data object is used for which purpose(s). For example, a typical privacy policy such as "we will collect and use customer identifiable information for billing purposes and to enable us to anticipate and resolve problems with your service" does not specify who can access the customer information, but only states that the information can be accessed for the purposes of billing, customer service and possibly some analysis. Another difficulty of privacy protection is that the comfort level of data usage varies from individual to individual. For example, some online consumers may feel that it is acceptable to disclose their purchase history or browsing habits in return for better service, such as site personalization [20]. Other customers, however, may believe that such techniques violate their privacy.

Observing these challenges, we believe that in order to protect data privacy, the notion of purpose must play a major role in access control models and that an appropriate metadata model must be developed to support such privacy-centric access control models. In this article, we address this goal by presenting a comprehensive approach to purpose management, which is the fundamental building block on which purpose-based access control can be developed. Our approach is based on intended purposes, which specify the intended usage of data, and access purposes, which specify the purposes for which a given data element is accessed. Both intended and access purposes are specified with respect to a hierarchical structure that organizes a set of purposes for a given enterprise. A key feature of our proposed model is that it also supports explicit prohibitions, thus allowing privacy officers to specify that data should not be used for a given set of purposes. We also formally define the notion of purpose compliance, which is the basis for verifying that the purpose of a data access complies with the intended purposes of the data.

An important issue that we also address in this article is the granularity of data labeling, i.e., the units of data with which purposes can be associated. We address this issue in the context of relational databases and propose four different labeling schemes, each providing a different granularity. Using our approach it is thus possible to associate a purpose (or a set of purposes) with an entire table, with each column within a table with each tuple within a table or with each attribute within a tuple. We also propose an approach to represent purpose information, which results in low storage overhead. Furthermore, we exploit query modification techniques to support data filtering based on purpose information. Such techniques ensure efficient query processing even in the case of fine-grained purpose labeling.

Another key contribution of our work is that we address the problem of how to determine the purpose for which certain data are accessed by a given user. We believe that this issue may be satisfactorily addressed by relying on role-based access control (RBAC) models [2,14,15,27]. However, in order to support policies specifying for which purpose a certain data can be accessed by a given role, we expand conventional RBAC models with the notion of conditional role which is based on the notions of role and system attribute.

The remainder of this article is organized as follows. In Sect. 2, we provide a brief overview for privacy related technologies available today. We formally define the notion of purpose in Sect. 3 and describe our purpose labeling model in Sect. 5. We introduce the notion of conditional role and present a method for determining access purposes in Sect. 4. In Sect. 6 we present our implementation, and we report experimental results in Sect. 7. We suggest future work and conclude our discussion in Sect. 8.

## 2 Related work

Our work is related to several topics in the area of privacy and security for data management, namely privacy policy specification, privacy-preserving data management systems and multilevel secure database systems. We now briefly survey the most relevant approaches in these areas and point out the differences of our work with respect to these approaches.

The W3C's platform for privacy preference (P3P) [31] is an industry standard that intends to provide an automated method for users to gain control over the use of their personal information collected by the web sites they visit. P3P allows web sites to encode their privacy practice in a machine-readable format, such as what data is collected, who can access the data for what purposes and how long the data will be stored by the sites. P3P enabled browsers can read this privacy policy automatically and compare it to the consumer's set of privacy preferences which are specified in a privacy preference language such as a P3P preference exchange language (APPEL) [30], also designed by the W3C.

Even though P3P provides a standard means for enterprises to make privacy promises to their users, P3P does not provide any mechanism to ensure that these promises are consistent with the internal data processing. Thus, P3P is merely a tool for making promises and does not help enterprises to keep their promises. Note that publishing an attractive P3P policy without any adequate enforcement mechanism may put an enterprise at risk of reputation damage and potential lawsuits.

The enterprise privacy authorization language (EPAL) [17] proposed by IBM is a formal language for writing enterprise privacy policies to govern data handling practices in IT systems. An EPAL policy defines hierarchies of data-

categories, user-categories and purposes. User-categories are the entities (users/groups) that use collected data, and data-categories define different categories of collected data. Purposes model the services for which data is intended to be used. An EPAL policy also defines sets of actions, obligations and conditions. Actions model how the data is used (e.g., read or write), and obligations define actions that must be taken by the environment of EPAL. Lastly, conditions are boolean expressions that evaluate the context. Privacy authorization rules are defined using these elements, and each rule allows or denies actions on data-categories by user-categories for certain purposes under certain conditions while mandating certain obligations.

While providing a language for specifying policies on data categories, EPAL does not provide support for linking the data categories with data stored in databases. Nor does the EPAL work address the issue of how to efficiently enforce these policies when data is accessed.

Previous work on multilevel secure relational databases [6,9,18,25] also provides many valuable insights for designing a fine-grained secure data model. In a multilevel relational database system, every piece of information is classified into a security level, and every user is assigned a security clearance. The system ensures that each user gains access to only the data for which he has proper clearance, according to the well known basic restrictions [5]. These constraints ensure that there is no information flow from a higher security level to a lower security level and that subjects with different clearances see different versions of multilevel relations.

A major difference of our approach with respect to multilevel secure databases is that in our approach each data element is associated with set of purposes, as opposed to a single security level. Also, the purposes form a hierarchy and can vary dynamically. These requirements are more complex than those concerning traditional multilevel secure applications. On the other hand, we are not concerned with information flow issues in this article.

The concept of Hippocratic databases, incorporating privacy protection within relational database systems, was introduced by Agrawal et al. [1]. The proposed architecture uses privacy metadata, which consist of privacy policies and privacy authorizations stored in two tables. A privacy policy defines for each attribute of a table the usage purpose(s), the external-recipients and retention period, while a privacy authorization defines which purposes each user is authorized to use.

Lefevre et al. [22] present an approach to enforcing privacy policy in database environments. Their work focuses on ensuring limited data disclosure based on the premise that data providers (i.e., the subjects about whom the data is stored) have control over who is allowed to see their personal data and for what purpose. In their work, they introduce two models of cell-level limited disclosure enforcement (table and query semantics) and suggest an implementation based on query modification techniques.

Although the work on the Hippocratic databases [1,22] is closely related to ours, our approach has some notable differences. First, we introduce more sophisticated concepts of purpose, i.e., purposes are organized in a hierarchy. Our experimental result shows that our approach, even though more sophisticated, does not increase the complexity of data management, nor does it introduce much overhead. The second difference is that we support the explicit prohibition of purpose and the association of a set of purposes with a data element, which their approach does not provide. Third, we provide a comprehensive framework for purpose and data management, which are not considered in their work.

Role-based access control [2,14,15], which has made a significant impact on many access control systems, greatly simplifies the specification and management of security policies within an enterprise. The basic concept of RBAC is as follows: permissions are assigned to functional roles within an enterprise and individual users are then authorized to the necessary permissions by being assigned to a role or a set of roles. Most RBAC models also include a role hierarchy, a partial order defining a relationship between roles and to facilitate the administration tasks.

The idea of incorporating attributes into RBAC models has been proposed to address some distinct problems in RBAC. Chen et al. [8] introduced the attributes associated with roles in order to enforce global constraints such as the principle of separation of duty. They discuss various attributes for roles, permissions, users and sessions and suggest a practical way to specify and enforce constraints based on these attributes. The notion of role attribute is also presented in [16,21] to provide more flexibility to the access control in RBAC. In [21], Kumar et al. introduced the notion of role context (i.e., role attributes) and context filter (i.e., constraints) to limit the applicability of the role's of permissions to a subset of the target objects. Goh et al. [16] discusses various constraints which utilize role attributes; for instance, conditional activation and deactivation of roles and qualification for role membership.

Our notion of role attributes is closely related to the notions presented in [16,21] in that it enables the specification and enforcement of context-sensitive policies in RBAC. However, we build upon and further elaborate the existing notions with the presence of role hierarchy to achieve fine-grained administrative control in RBAC by incorporating logical constraints [4].

## 3 Purpose

As previously mentioned, a privacy policy mainly concerns with which data object is used for which purpose(s).

Consequently, purpose is a central concept in many privacy protecting access control models [1,3,17,22]. However, the concept of purpose has not yet been thoroughly investigated. In this section, we formally define the notion of purpose and discuss key related issues.

### 3.1 Definition of purpose

In order to preserve the privacy of data providers, every data access must comply with the privacy policies on which data providers have agreed. A typical privacy policy for a data element includes purpose(s), retention, condition and obligation. It states that the particular data element can be accessed only for the specific purpose(s) on the specific condition. The retention indicates how long the data element can be retained, and the obligation designates the actions that must be followed after an access to the data element is allowed. The aspect that is most interesting to us is the purpose, as the purpose directly dictates how accesses to data items should be controlled. P3P defines purpose as "the reason(s) for data collection and use" and specifies a set of purposes, including *current, admin, develop, contact, telemarketing* [31]. In common business environments, however, purposes naturally have a hierarchical relationships among them, i.e., generalization and specialization relationships. For instance, a group of purposes such as *direct-marketing* and *third-party marketing* can be represented by a more general purpose, *marketing*. This suggests that purposes can be organized according to the hierarchical relationships to simplify the management of purposes. The next definition formalizes the above discussion.

**Definition 1** (purpose and purpose tree) *A purpose describes the reason(s) for data collection and data access. A set of purposes, denoted as $\mathcal{P}$, is organized in a tree structure, referred to as* purpose tree *and denoted as $\mathcal{PT}$, where each node represents a purpose in $\mathcal{P}$ and each edge represents a hierarchical relation (i.e., specialization and generalization) between two purposes. Let $p_i$, $p_j$ be two purposes in $\mathcal{PT}$. We say that $p_i$ is an ancestor of $p_j$ (or $p_j$ is a descendent of $p_i$) if there exists a downward path from $p_i$ to $p_j$ in $\mathcal{PT}$.*

Figure 1 gives an example of purpose tree. One can argue that it would be more advantageous to organize purposes according to a partial order relation (e.g., directed acyclic graph) instead of a tree, as it would allow a single node to associate with multiple parent nodes. However, as we show in Sect. 6, a tree structure allows an efficient design for storing and processing purposes. The following notations will be used throughout this article.

**Notations** (*ancestors and descendants*) Let $\mathcal{PT}$ be a purpose tree and $\mathcal{P}$ be the set of purposes in $\mathcal{PT}$. Let $P \subseteq \mathcal{P}$ be a set of purposes in $\mathcal{PT}$.

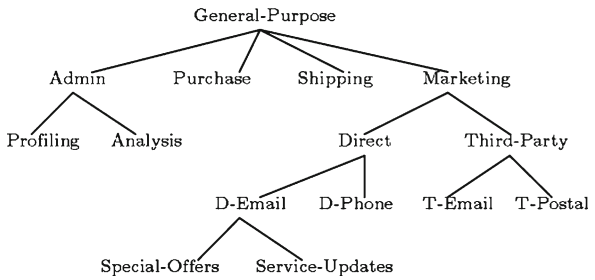



**Fig. 1** Purpose tree

- Ancestors($P$), denoted by $P^{\uparrow}$, is the set of all nodes that are ancestors of nodes in $P$, including nodes in $P$ themselves.
- Descendants($P$), denoted by $P^{\downarrow}$, is the set of all nodes that are descendants of nodes in $P$, including nodes in $P$ themselves.
- We use $P^{\updownarrow}$ to denote the set of all nodes that are either ancestors or descendants of nodes in $P$, that is, $P^{\updownarrow} = P^{\uparrow} \cup P^{\downarrow}$

*Example* 1 Let $\mathcal{PT}$ be the purpose tree in Fig. 1.

1. Ancestors(*Analysis*) = $(Analysis)^{\uparrow}$ = {*Analysis*, *Admin*, *General-Purpose*}
2. Descendants(*Third-Party*) = $(Third\text{-}Party)^{\downarrow}$ = {*Third-Party*, *T-Email*, *T-Postal*}

Intuitively, an access to a specific data element should be allowed if the allowed purposes for the data, stated by the privacy policies, include or imply the purpose of the data access. We refer to purposes associated with data and thus regulating data accesses as *intended purposes*, and to purposes for accessing data as *access purposes*. Intended purposes can be viewed as brief summaries of privacy policies for data, stating for which purposes data can be accessed. When an access to a data item is requested, the access purpose is checked against the intended purposes for the data item.

Most privacy policies and privacy preferences are permissive in nature in that they selectively allow data access for a set of purposes. Thus, the absence of a particular purpose from the set of allowed purposes is interpreted as that data access for the purpose is not allowed. This concept is adopted in privacy policy languages such as P3P [31]. However, some privacy policies may explicitly prohibit access to data for certain purposes. For example, suppose that in order to comply with COPPA [11], a company decides not to use any information about children of age under 13 for the *marketing* purpose. This policy is prohibitive in nature as it explicitly disallows access to the data items belonging to minors for the particular purpose.

Our design of intended purposes supports both permissive and prohibitive privacy policies. An intended purpose consists of two components: *allowed intended purposes* (AIP for short) and *prohibited intended purposes* (PIP for short). This structure allows more compact and flexible policies in our model. Moreover, by using PIP, we can guarantee that data accesses for particular purposes are never allowed. Conflicts between the AIP and the PIP for the same data element are resolved by applying the denial-takes-precedence policy where PIP overrides AIP.

**Definition 2** (intended purpose) *Let $\mathcal{PT}$ be a purpose tree and $\mathcal{P}$ be the set of purposes in $\mathcal{PT}$. An intended purpose IP is a tuple $\langle AIP, PIP\rangle$, where $AIP \subseteq \mathcal{P}$ and $PIP \subseteq \mathcal{P}$ are two sets of purposes. AIP is called the set of allowed intended purposes and PIP the set of prohibited intended purposes. The set of purposes implied by IP, denoted by IP*, is defined to be $AIP^{\downarrow} - PIP^{\updownarrow}$.*

*Example* 2 Suppose $IP = \langle\{Admin, Direct\}, \{D\text{-}Email\}\rangle$ is defined over the purpose tree given in Fig. 1.

1. $AIP^{\downarrow} = (Admin)^{\downarrow} \cup (Direct)^{\downarrow} = \{Admin, Profiling, Analysis, Direct, D\text{-}Email, D\text{-}Phone, Special\text{-}Offers, Service\text{-}Updates\}$.
2. $PIP^{\updownarrow} = (D\text{-}Email)^{\downarrow} \cup (D\text{-}Email)^{\uparrow} = \{D\text{-}Email, Special\text{-}Offers, Service\text{-}Updates, D\text{-}Email, Direct, Marketing, General\text{-}Purpose\}$.
3. $IP^* = AIP^{\downarrow} - PIP^{\updownarrow} = \{Admin, Profiling, Analysis\}$.

We note that the use of both AIP and PIP is not strictly necessary. In fact, $IP = \langle AIP, PIP\rangle$ can be always transformed into $IP' = \langle AIP', \emptyset\rangle$, where $AIP' = AIP^{\downarrow} - PIP^{\updownarrow}$. IP and IP′ are semantically equivalent. However, we decided to use both AIP and PIP in our model for the following reasons. First, as previously mentioned, some privacy policies are naturally permissive whereas some are naturally prohibitive. A direct support for both types of policies is valuable as it minimizes the possibility of implementation errors. Second, using PIP as exceptions, IP can be expressed in a more compact manner. For instance, suppose a privacy policy allows data access for any purpose except for *Third-Party* in Fig. 1. This can be simply expressed as $IP = \langle\{General\text{-}Purpose\}, \{Third\text{-}Party\}\rangle$ using both AIP and PIP while using AIP only, $IP = \langle\{Admin, Purchase, Shipping, Direct\}, \emptyset\rangle$. Lastly, using PIP one can make sure that data access for particular purpose(s) is never allowed. This guarantee is often required for organizations who want to keep their data management practice in compliance with privacy laws.

An access purpose is the purpose of a particular data access, which is determined or validated by the system when the data access is requested. How to determine or verify access purposes is not trivial, and this issue has not been thoroughly investigated in previously proposed models. We discuss this issue in detail in Sect. 4.

As already discussed, an access decision is made based on the relationship between the access purpose and the intended purpose of data. That is, an access is allowed only if the access purpose is included in the implication of the intended purpose; in this case we say the access purpose is *compliant* with the intended purpose. The access is denied if the implication of the intended purpose does not include the access purpose; we then say that the access purpose is *not compliant* with the intended purpose.

**Definition 3** (access purpose compliance) *Let $\mathcal{PT}$ be a purpose tree. Let $IP = \langle AIP, PIP\rangle$ and AP be an intended purpose and an access purpose defined over $\mathcal{PT}$, respectively. AP is said to be compliant with IP according to $\mathcal{PT}$, denoted as $AP \Leftarrow_{\mathcal{PT}} IP$, if and only if $AP \in IP^*$.*

*Example* 3 Let $\mathcal{PT}$ be the purpose tree in Fig. 1, and let *IP* and *AP* be an intended purpose and an access purpose defined based on $\mathcal{PT}$, respectively.

1. Suppose $IP = \langle\{General\text{-}Purpose\}, \{Third\text{-}Party\}\rangle$. If $AP = Marketing$, then $AP \Leftarrow_{\mathcal{PT}} IP$ as $Marketing \in PIP^{\updownarrow}$. However, if $AP = Admin$, then $AP \Leftarrow_{\mathcal{PT}} IP$, as $Marketing \notin PIP^{\updownarrow}$ and $Marketing \in AIP^{\downarrow}$.
2. Suppose $IP = \langle\{Admin, Purchase, Shipping\}, \{General\text{-}Purpose\}\rangle$. Then no *AP* defined over $\mathcal{PT}$ is compliant with *IP*.
3. Suppose $IP = \langle\{General\text{-}Purpose\}, \emptyset\rangle$. Any *AP* defined over $\mathcal{PT}$ is compliant with *IP*.

### 3.2 Intended purpose management

Based on the purpose tree, an intended purpose is specified for each data element according to the privacy policy on which the data provider has agreed. We assume that the organization has already established a set of comprehensive privacy policies which are compliant with existing privacy laws and that, as a part of the data collection process, data providers are informed of and agree to the privacy policies via some mechanism such as P3P [31]. Under this assumption, the privacy policy concerning each data element is predetermined; consequently, the intended purposes of most data items are predetermined.

We use the phrase "most data items" deliberately as in some exceptional cases the intended purpose for certain data can vary depending on individual data providers. For instance, enterprises who wish to fully comply with COPPA should have a different set of intended purposes concerning data collected from children under 13. Another example can be found in enterprises whose business activities span multiple nations. As different regions have different privacy requirements, such enterprises must carefully differentiate

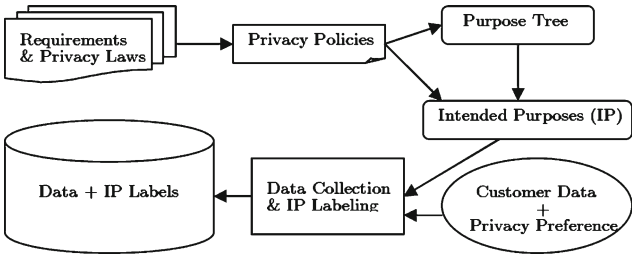



**Fig. 2** Intended purpose management process

data providers from various regions and apply appropriate privacy policies (e.g., a set of intended purposes) to the collected data. The intended purposes can also vary due to additional options provided by enterprises. A common example of these options is the enrollment of a mailing-list. Whether or not the email address of a particular customer can be used for sending information about new services is dependent on the explicit decision of the customer. Note that the intended purposes can be further individualized upon the requests from data providers after data collection as well. The overall organization of the purpose management process is illustrated in Fig. 2.

*Example* 4 Suppose a company has established the following privacy policies.

- We use your information for purchasing purposes, to provide services to you and to inform you of services that may better meet your needs.
- We will not disclose your information to third parties who want to market products to you unless you allow us to do so.
- We do not use information of children under 13 for any purpose other than providing requested services.
- The web server administrators may collect some data such as your IP address, referrer and your web browser information. We do not make use of this information, but it may be used by system administrators to provide better service to you.

Table 1 illustrates the intended purposes for the data collected by the company, based on the privacy policies above and the purpose tree in Fig. 1. Group 1 represents customers who are not children and have given consents for third-party marketing, and Group 2 represents customers who are not children and have not given consents for third-party marketing.

## 4 Access purpose determination

An access purpose is the reason for accessing a data item, and it must be determined by the system when a data access is requested. Evidently, how the system determines the purpose of an access request is crucial as the decision of whether or not the access should be allowed is directly based upon the access purpose. In this section we present a possible method for determining access purposes.

**Table 1** Predetermined intended purposes

| | Under 13 | Group 1 | Group 2 |
|---|---|---|---|
| Name | $\langle\{G\},\{A,M\}\rangle$ | $\langle\{G\},\emptyset\rangle$ | $\langle\{G\},\{T\}\rangle$ |
| Address | $\langle\{G\},\{A,M\}\rangle$ | $\langle\{G\},\emptyset\rangle$ | $\langle\{G\},\{T\}\rangle$ |
| Phone | $\langle\{G\},\{A,M\}\rangle$ | $\langle\{G\},\emptyset\rangle$ | $\langle\{G\},\{T\}\rangle$ |
| Order-history | $\langle\{G\},\{A,M\}\rangle$ | $\langle\{G\},\emptyset\rangle$ | $\langle\{G\},\{T\}\rangle$ |
| Web-log | $\langle\{A\},\{A,M\}\rangle$ | $\langle\{A\},\emptyset\rangle$ | $\langle\{A\},\{T\}\rangle$ |

*G* general, *A* admin, *P* purchase, *S* shipping, *M* marketing, *T* third-party

### 4.1 Preliminaries

There are various possible strategies to determine access purpose. First, the users can be required to state their access purpose(s) along with the requests for data access. Even though this method is simple and can be easily implemented, it requires complete trust on the users and the overall privacy that the system is able to provide entirely relies on the users' trustworthiness. Another possible method is to register each application or stored-procedure with an access purpose. As applications or stored-procedures have limited capabilities and can perform only specific tasks, it can be ensured that data users use them to carry out only certain actions with the associated access purpose. This method, however, cannot be used for complex stored-procedures or applications as they may access various data for multiple purposes. Lastly, the access purposes can be dynamically determined by the system, based on the current context. For example, suppose an employee in the shipping department is requesting to access the address of a customer by using a particular application in a normal business hour. From this context (i.e., the job function, role, the nature of data to be accessed, the application identification and the time of the request), the system can reasonably infer that the purpose of the data access must be shipping. The key challenge for implementing this method is that it may be difficult to infer the access purposes both accurately and efficiently.

In this article, we utilize the first method where the users are required to explicitly state their access purpose(s) when they try to access data. That is, the users provide an access purpose for each query they issue. To make our discussion more concrete, we extend SQL queries as well as update commands by adding an additional clause for access purposes.[1] For instance, a simple select statement "**SELECT**

[1] We are currently investigating approaches that can be implemented on top of commercial DBMS without requiring any extension to the DBMS internal or to SQL.

name **FROM** customer" is extended to a form of "**SELECT** name **FROM** customer **FOR** marketing". Our proposed SQL extensions are provided in the Appendix. We note that this extension does not change the underlying semantics of SQL. As our approach relies on query modification, the purpose information provided by the FOR clause in a query is used to determine how to modify the query to enforce purpose-based access control policies, and the resulting query is a standard SQL query. We provide more detailed discussion on our approach and query modification in Sect. 6.

We further extend this approach by adding a validation process which verifies the stated access purposes. That is, users are required to state their access purposes along with their queries, and the system validates the stated access purposes by ensuring that the users are indeed allowed to access data for the particular purposes.

To ease the management of access purpose authorizations, users are granted authorizations through their roles, i.e., access purpose authorizations are granted to roles, not directly to individual users. This method has a great deployment advantage as many systems are already using RBAC mechanisms for the management of access permissions. This approach is also reasonable as access purposes can be granted to the tasks or functionalities over which roles are defined within an organization. However, using an RBAC mechanism for the management of both access permissions and access purposes may increase the complexity of the role engineering tasks. To address this problem, we introduce a simple extension to RBAC, which simplifies the role administration and also provides increased flexibility.

In this section, we first present our extended RBAC model and discuss the details of the access purpose authorization and verification based on this model. We do not discuss the general concepts of RBAC, assuming that readers are already familiar with them. For interested readers, we refer to [2,14,15].

### 4.2 Role attributes, system attributes and conditional roles

Role hierarchies are considered a key component of RBAC models, and in this article, our extension also assumes hierarchical RBAC [2,15]. More specifically, we assume inverted tree hierarchies which are used to facilitate sharing of resources [26]. Figure 3 illustrates an inverted tree hierarchy in a hypothetical enterprise. In this hierarchy, the root of the hierarchy, "Employee", represents the most general role (i.e., the junior-most role), and a permission assigned to a role is inherited downward and becomes available to all of its senior roles.

As the role hierarchy is predefined primarily for the access permission assignments, it is possible that the existing role definitions do not adequately specify the set of users to whom we wish to grant an access purpose. For instance, consider

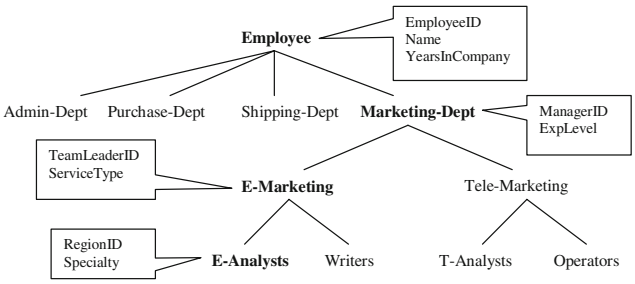



**Fig. 3** Role hierarchy and role attributes

the purpose tree in Fig. 1 and the role hierarchy in Fig. 3. Suppose that we wish to allow some users in the E-Marketing team to access data for the purpose of "Service-Update", under the "E-Marketing" role there are two descendant roles: "E-Analysts" and "Writers". "E-Analysts" are the users who analyze the customer information and prepare the contents of emails, and "Writers" are the users who write and send emails to customers. Note that these two roles are defined based on the access permission assignments in that the permissions to access the customer profiles are exclusively assigned to the "E-Analysts" role, while the permissions to access the email addresses of the customers are exclusively assigned to the "Writers" role. However, as we want to assign the access purpose only to the users who are responsible for the specific task of sending updated service information, neither the definition of "E-Analysts" nor "Writers" matches our intention. Moreover, assigning the access purpose to the "E-Marketing" role is not desirable as it will allow all the users with the "E-Marketing" role to access data with the access purpose. An alternative solution is to split the "E-Marketing" role or the "E-Analysts" and the "Writers" roles into more specific roles. However, this method requires the reconstruction of both the user assignments and the permission assignments for the modified roles. Authorizing the access purpose on an individual basis is not an elegant solution either, as it does not utilize the existing role hierarchy and thus is not scalable. In order to address these issues, we introduce a new concept, *conditional role*, which is based on the notion of *role attribute* and *system attribute*.

Role attributes are the pre-assigned, specific descriptions associated with each role. The role attributes of each role are defined by system administrators at the time of role creation, and each role inherits the role attributes that are defined at the ancestor roles. When a user is assigned to a role, the values of the role attributes (both defined and inherited role attributes) are specified according to the relevant information of the particular user. Then when the user activates the role, the values of the role attributes are loaded and made available to the access control system until the user deactivates the role. As such, the role attributes can be viewed as a cached user information that is relevant to the specific roles.

**Definition 4** (role attributes) *Let $\mathcal{R}$ be the set of roles defined in the system. Every role $r \in \mathcal{R}$ has a set of attributes, denoted by $r.Attributes = \{r.attr_1, \ldots, r.attr_n\}$, that are defined for $r$ or inherited from the ancestor roles of $r$. Each attribute $r.attr_i$ has a data type $\tau_i$ which determines the set of values that can be assigned to this attribute* [*called the domain of this attribute and denoted by* $\mathsf{domain}(\tau_i)$] *and the set of binary predicates that can be applied to values of this type* [*denoted by* $\mathsf{preds}(\tau_i)$]. *For example, if a data type $\tau_i$ is an ordered type, then* $\mathsf{preds}(\tau_i) = \{<, \leq, >, \geq, =, \neq\}$. *Each data type determines the interpretation of these predicates.*

*The values of the role attributes for a particular role $r$ are specified within the domains for each user when a user is assigned to $r$. The specified role attribute values of a user $u$ under $r$, denoted by $r(u).Attributes$, are available to the access control system from the time the user activates $r$ to the time the user deactivates $r$.*

Figure 3 shows an inverted tree role hierarchy and the role attributes. Role "Employee" has three role attributes: *EmployeeID*, *Name* and *YearsInCompany*, and the roles below "Employee" all inherit the role attributes of "Employee". For instance, the role attributes of the "Marketing-Dept" role include the role attributes of "Employee" as well as *ManagerID* and *YearsInDept*. Even though any role can be associated with a set of attributes, we only show the role attributes of a few selected roles in the figure for simplicity. We do not show the inherited role attributes for the same reason.

Using the role attributes we can assign an access purpose to a specific subset of users in the same role, that is, we can assign a particular access purpose to a role with a set of conditions that must be satisfied by the users of the role in order to access data with the access purpose. For instance, consider the previous scenario where we wish to assign the access purpose "Service-Update" to a particular group of users in the E-Marketing team. Provided that the attribute *ServiceType* is defined for the role "E-Marketing" as in Fig. 3, we first set the value of this attribute to, say, "Update-Info" only for the users who are responsible for the task. Then by assigning the access purpose to the "E-Marketing" role with a condition saying that the user's value of the attribute *ServiceType* must be equal to "Update-Info", we can authorize the access purpose "Service-Update" only to the users whom we wish to grant the access purpose.

When authorizing access purposes, one may also need to consider the states of the system where the authorizations should become effective (or ineffective). For instance, we may wish to allow an access purpose to be used by a set of users only in a specific time interval (e.g., business hours) or only when the users are logged into specific machines (e.g., machine identification number). Thus, the system information that affects the access purpose authorizations must be clearly defined and available to the access control system. We refer to this system information as system attributes.

**Definition 5** (system attributes) *Let $\mathcal{S}$ be a target system. Then $\mathcal{S}$ is described by a set of attributes, denoted by $\mathcal{S}.Attributes = \{\mathcal{S}.attr_1, \ldots, \mathcal{S}.attr_n\}$, and these attributes are available to the access control system at all times. Each attribute has a data type which determines the domain of the attribute and the set of applicable binary predicates. The system attributes are defined by system administrators for the application needs, and the values of the system attributes in a system state $s$, denoted by $\mathcal{S}(s).Attributes$, specify the environment of the system in the state $s$.*

Now we introduce the notion of conditional role which utilizes both the role attributes and the system attributes to describe a specific set of users in a particular system environment.

**Definition 6** (conditional roles) *Let $\mathcal{R}$ be the set of roles defined in the system $\mathcal{S}$. A conditional role $cr$ is defined as a pair $\langle r, C\rangle$, where $r \in \mathcal{R}$ and $C$ is a propositional logic formula that can be constructed from primitive constraints using $\wedge$ (AND), $\vee$ (OR) and $\neg$ (NOT). Each primitive constraint in $C$ is of the form $x\ \phi\ y$, where $x \in r.Attributes \cup \mathcal{S}.Attributes$, $y \in r.Attributes \cup \mathcal{S}.Attributes \cup \mathsf{domain}(\tau)$, where $\tau$ is the data type of $x$, and $\phi \in \mathsf{preds}(\tau)$. If $y$ is an attribute name in $r.Attributes \cup \mathcal{S}.Attributes$, then it must have the same type as $\tau$. We say that a user* u belongs to *a conditional role $cr_i = \langle r_i, C_i\rangle$ in a system state $s$ if and only if the following conditions are satisfied:*

1. *The user $u$ has activated a role $r$ that dominates the role $r_i$.*
2. *When attribute names in $C_i$ are replaced with the corresponding values in $r(u).Attributes$ and $\mathcal{S}(s).Attributes$, the constraint $C_i$ evaluates to true. The logical operators $\wedge$, $\vee$ and $\neg$ are evaluated using their standard semantics.*

More specifically, if $r = \mathrm{r}_i$ and the second condition is satisfied, we say that *$r$ explicitly belongs to $cr_i$*. On the other hand, if $r \in \mathrm{Descendants}(\mathrm{r}_i)$, but $r \neq \mathrm{r}_i$, and the second condition is satisfied, then we say that *$r$ implicitly belongs to* $\mathrm{cr}_i$.

### 4.3 Access purpose authorization and verification

As discussed in the previous section, access purposes are authorized to users through conditional roles. The use of conditional roles provides great flexibility in that the authorizations are sensitive to both the user profiles and the system environments. In this section, we formally define the access purpose authorization and its verification.

**Definition 7** (access purpose authorization) *Let $\mathcal{PT}$ be a purpose tree, $\mathcal{P}$ be the set of purposes in $\mathcal{PT}$ and $\mathcal{R}$ be the set of roles defined in the system $\mathcal{S}$. An access purpose is authorized to a specific set of users by a pair* ⟨ap, cr⟩, *where* ap ∈ $\mathcal{P}$ *and* cr *is a conditional role defined over $\mathcal{R}$ and $\mathcal{S}$.*

Note that both the access purpose and the conditional roles are organized in hierarchies. Consequently, an access purpose authorization has its implications, i.e., authorizing an access purpose *ap* for a conditional role *cr* implies that the users belonging to *cr* either explicitly or implicitly are authorized to access data with *ap* as well as all the descendants of *ap* in the purpose tree. The definition of access purpose verification below captures these implications of access purpose authorizations.

**Definition 8** (access purpose verification) *Let $\mathcal{PT}$ be a purpose tree and $\mathcal{P}$ be the set of purposes defined over $\mathcal{PT}$. Let $\mathcal{R}$ be the set of roles defined in the system $\mathcal{S}$. Given an access purpose* ap *and a role* r *activated by a user* u, ap *is valid for* u *under* r *if there exists an access purpose authorization $\langle ap_i, cr_j \rangle$, where* $\text{ap}_i \in \mathcal{P}$ *and* $\text{cr}_j = \langle r_j, C_j \rangle$ *is a conditional role defined over $\mathcal{R}$ and $\mathcal{S}$, satisfying the following two conditions*:

1. ap ∈ *Descendants*($ap_i$).
2. *The user* u *belongs to the conditional role* $\text{cr}_j$ *either explicitly or implicitly.*

For example, the access purpose "Service-Update" in the previous scenario can be assigned to ⟨E-Marketing, (ServiceType = "Update-Info") ∧ (timeofday ≥ 9) ∧ (timeofday ≤ 17)⟩, assuming that *timeofday* is defined as a system attribute. Then only the users who activate the role "E-Marketing" (or the two descendant roles) with their *ServiceType* attribute equal to "Update-Info" can access data with the purpose of "Service-Update" between 9 am and 5 pm.

## 5 Data labeling model

In order to build an access control model based on the notion of purpose, we must consider a specific data model and devise a proper labeling scheme based on the model. One question here is how intended purposes are associated with data. More specifically, we have to determine at what level of granularity data will be associated with intended purposes. Under relational data models, intended purpose can be assigned to every relation, to every tuple in every relation, to every attribute in every relation or to every data element in every relation. In this section, we discuss this issue of data granularity.

Data providers (e.g., customers) are usually reluctant to allow any use of their information unless it is absolutely necessary. At the same time, data users (e.g., enterprises) want to make use of the collected data for the necessary tasks as well as other tasks such as analysis and marketing. Consequently, some form of negotiating process may occur between these two parties through opt-in/opt-out procedures. Note that the comfort level of privacy can considerably vary from individual to individual.

**Table 2** Customer table

| c_id | c_id_ip | Name | Name_ip | Income | Income_ip |
|---|---|---|---|---|---|
| 1001 | ⟨{G},∅⟩ | John | ⟨{G},{M}⟩ | 110,000 | ⟨{A},{M}⟩ |
| 1002 | ⟨{G},∅⟩ | Paul | ⟨{G},∅⟩ | 56,000 | ⟨{G},∅⟩ |
| 1003 | ⟨{G},∅⟩ | Jack | ⟨{G},∅⟩ | 48,000 | ⟨{G},{T}⟩ |

*G* general, *A* admin, *M* marketing, *T* third-party

Consider a data category *purchase history*, which consists of *name*, *financial-info*, *product*, and *purchase-date*. As this category includes sensitive information such as *financial-info*, many customers would not want to allow any use of the information in this category. However, information such as *name*, *product*, and *purchase-date* can be valuable for enterprises as they can use such information for analyzing their sale patterns or profiling customers. It is obvious that in order to make the best use of data while at the same time ensure that data providers feel comfortable, the granularity of the data labeling model must be fine. Thus, the labeling model should allow the assignments of intended purposes with data at the most fine-grained level. That is, we should be able to assign an intended purpose to each data element in every tuple, e.g., for each attribute and for each data provider (see Table 2).

However, this most fine-grained approach is not always necessary. For instance, some data naturally have a hierarchical structure, in which the addresses of customers are stored in a relation that consists of *street*, *city*, *state*, and *Zip-code*. Typically, a customer allows or prohibits access to the entire address, not to the individual sub-elements. Thus, it is not necessary to associate each data element with an intended purpose because labeling at the element-level granularity would result in storing an identical intended purpose for every data element redundantly. However, intended purpose for the address can vary depending on each individual. To address these concerns, the labeling model should allow the assignment of intended purpose to each tuple of a relation (see Table 3).

**Table 3** Address table

| c_id | Street | City | State | Zip_code | Addr_ip |
|---|---|---|---|---|---|
| 1001 | 32 Oval dr | Lafayette | IN | 47907 | ⟨{G},{A,M}⟩ |
| 1002 | 433 State rd | Chicago | IL | 46464 | ⟨{G},∅⟩ |
| 1003 | 199 First ave | Boston | CA | 02139 | ⟨{G},{T}⟩ |

*G* general, *A* admin, *M* marketing, *T* third-party

**Table 4** Order table

| or_id | c_id | Product | Credit_info | Date | Status |
|---|---|---|---|---|---|
| 101 | 1001 | P303 | V3434-343-2222 | 10/23/03 | shipped |
| 102 | 1002 | P887 | V5675-374-5892 | 07/20/04 | packaged |
| 103 | 1003 | S99-6 | M6584-677-4911 | 08/22/04 | ordered |

**Table 5** Privacy-policy table

| Table_name | Column_name | ip |
|---|---|---|
| Order | Product | ⟨{A, P, S}, ∅ ⟩ |
| Order | Credit_info | ⟨{P}, {M}⟩ |
| Order | Date | ⟨{A, P, S}, {M}⟩ |
| Order | Status | ⟨{A, P, S}, ∅ ⟩ |
| Access_log | ALL | ⟨{A, P}, ∅ ⟩ |

*A* admin, *P* purchase, *S* shipping, M marketing

Another case we should consider is that there exists some information for which corresponding privacy policies are mandated by enterprises or by laws, i.e., data providers do not have a choice to opt-out from the required intended purposes. An example is the *Order* information in Table 4. As such information must be accessed to perform necessary tasks, enterprises can choose not to give customers any option to change the privacy policies governing this information. In such cases, the data elements in each column in the relation have the identical intended purpose. Thus, in order to avoid any redundancy, intended purposes should be assigned to each attribute of a relation using a *privacy policy table* (see Table 5). It is also possible that the intended purposes of every attribute in a relation be identical. Such cases occur when information in a relation is meaningful as a whole tuple, but individual elements or tuples do not have any usefulness. The *access-log* table in Table 7 is one such relation. In this case, the intended purposes are assigned to the entire relation by using a single entry in the privacy policy table (see Table 5).

Now we formally define our labeling model which incorporates the above discussion.

**Definition 9** (relation, attribute, tuple, and Element) *As in the standard relational model, data are stored in relations. A relation is characterized by the following two components.*

1. *A state-invariant relation scheme* $\mathcal{R}(A_1, \ldots, A_n)$*, where* $\mathcal{R}$ *is the name of the relation and each* $A_i$ *is an attribute over some domain* $D_i$*. Attributes*$(\mathcal{R})$ *denote the set of names of attributes in* $\mathcal{R}$*.*
2. *A state-dependent relation instance* $r_i$ *over* $\mathcal{R}$ *composed of distinct tuples of the form* $(a_1, \ldots, a_n)$*, where each element* $a_i$ *is a value in domain* $D_i$*.*

**Definition 10** (intended purpose labeling) *Let* $\mathcal{PT}$ *be a purpose tree and* $\mathcal{P}$ *be the set of purposes in* $\mathcal{PT}$*. Let IP be a set of all possible intended purposes defined over* $\mathcal{P}$ *and* $\mathcal{R}(A_1, \ldots, A_n)$ *be a relation. In our data labeling model, intended purposes are associated with* $\mathcal{R}$ *according to one of the following methods.*

1. Relation-based: *A relation-based labeling is a pair* $\langle \mathcal{R}, ip \rangle$*, where* $ip \in IP$*. Access to any data element in instances of* $\mathcal{R}$ *is governed by* $ip$*.*
2. Attribute-based: *An attribute-based labeling is a set* $\{\langle A_i, ip_i \rangle \mid A_i \in Attributes(\mathcal{R}) \wedge ip_i \in IP\}$*. Access to data element* $a_i$ *in any instance of* $\mathcal{R}$ *is governed by* $ip_i$*.*
3. Tuple-based: *A tuple-based labeling is a relation scheme* $\mathrm{Rtl}(A_1, \ldots, A_n, \ell)$*, where* $\ell$ *is a column having IP for its domain, such that* $\mathcal{R} = \prod_{A_1,\ldots,A_n}(\mathrm{Rtl})$*. Access to any data element in the* j*th tuple in any instance of* $\mathcal{R}$ *is governed by* $\ell_j$*.*
4. Element-based: *An element-based labeling is a relation scheme* $\mathrm{Rel}(A_1, \ell_1, \ldots, A_n, \ell_n)$*, where* $\ell_i (i = 1, \ldots, n)$ *is a column having IP for its domain, such that* $\mathcal{R} = \prod_{A_1,\ldots,A_n}(\mathrm{Rel})$*. Access to data element* $a_i$ *in any instance of* $\mathcal{R}$ *is governed by* $\ell_i$*.*

**Table 6** Access-log table

| Client_ip | Date | Time | Requested_url |
|---|---|---|---|
| 4.33.163.99 | 15/08/04 | 18:35:22 | /sci-fi/books/index.html |
| 218.232.444.33 | 15/08/04 | 19:35:53 | /home.html |
| 63.344.343.75 | 15/08/04 | 19:36:02 | /kids/music/index.html |

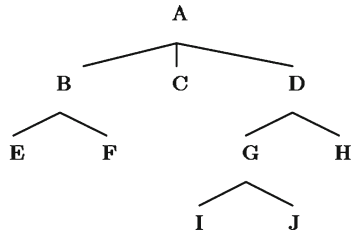
The first two types of labeling are intensional labeling schemes as they are defined at schema level. On the other hand, the third and fourth types are extensional labeling schemes as they are associated with data elements inside relation extensions.

The element-based labeling scheme is illustrated in Table 2, where each data element is labeled with an intended purpose. Table 3 is an example of the tuple-based labeling scheme, and here intended purposes are associated with each tuple. Tables 4 and 6, together with the privacy policy in Table 5, illustrate the attribute- and relation-based labeling schemes. Note that these tables only represent a conceptual view.[2] We discuss actual implementation strategies in Sect. 6.

## 6 Implementation

In this section we discuss two important implementation issues. The first is related to approaches for storing purpose

[2] The tables are represented as nested relations to make our examples more compact and intuitive. However, our implementation does not necessarily require nested relations.




(i) Purpose Tree

| p_id | p_name | parent | code | aip_code | pip_code |
|---|---|---|---|---|---|
| 1 | A | - | 0x200 | 0x3FF | 0x3FF |
| 2 | B | 1 | 0x100 | 0x130 | 0x330 |
| 3 | C | 1 | 0x080 | 0x080 | 0x280 |
| 4 | D | 1 | 0x040 | 0x04F | 0x24F |
| 5 | E | 2 | 0x020 | 0x020 | 0x320 |
| 6 | F | 2 | 0x010 | 0x010 | 0x310 |
| 7 | G | 4 | 0x008 | 0x00B | 0x24B |
| 8 | H | 4 | 0x004 | 0x004 | 0x244 |
| 9 | I | 7 | 0x002 | 0x002 | 0x24A |
| 10 | J | 7 | 0x001 | 0x001 | 0x249 |

(ii) pt_table

**Fig. 4** Purpose tree storage

information and for recording which purposes are associated with which data elements. The other issue is related to query modification techniques supporting data filtering with respect to the notion of purpose.

### 6.1 Privacy metadata storage

For both storage and performance efficiency, purposes are encoded as bit strings. Consider the purpose tree in Fig. 4(i). This purpose tree is encoded into a relation *pt_table* as shown in Fig. 4(ii). The first column *p_id* represents the identification number of each purpose node, which is determined according to the breadth-first search order[3] of the tree. The second column *p_name* represents the name of each purpose node, and the third column *parent* is used to capture the hierarchical relationships among the purpose nodes. The column *code* is the binary encoding of each purpose. For instance, in Fig. 4 the purpose *A* is encoded as '0 × 200' in hexadecimal representation, while the purpose *D* is encoded as '0 × 040' in hexadecimal form. Note that a binary representation provides significant advantages in that it is efficient in terms of storage and computation.

The last two columns *aip_code* and *pip_code* are pre-calculated encodings of purpose implications. As described in Sect. 3, when a purpose $p_i$ is used as an AIP, it implies that every descendant of $p_i$, including $p_i$ itself, is allowed. For instance, the purpose *B* in Fig. 4(i) used as an AIP implies that access is allowed for the purpose of *B* as well as *E* and *F*. Thus, the aip_code of *B* contains the implied set of *B*, which is the sum of the encodings of *B*, *E* and *F*. The pip_code of a particular purpose $p_i$ is computed similarly by summing the encodings of every descendant and ancestor of $p_i$ with the encoding of $p_i$ itself. Note that the last three columns of the *pt_table* can be automatically generated based on the first three columns using a simple procedure.

The other type of metadata to be stored is the intended purposes for the actual data. As described in Sect. 5, data elements are associated with intended purposes according to one of the intended purpose labeling schemes. When using the element-based labeling scheme, a table with $n$ columns is extended to $(n + 2n)$ columns;[4] $n$ data columns plus $2n$ columns for AIP and PIP. When using the tuple-based labeling scheme, a table with $n$ columns is extended to $(n + 2)$ columns; $n$ data columns plus 2 columns for AIP and PIP. While the element- and tuple-based labeling schemes require extensions to data tables, the attribute-based labeling scheme for a table with $n$ columns requires $n$ entries in the privacy policy table as shown in Table 5. Similarly, the relation-based labeling scheme for a table requires a single entry in the privacy policy table.

Intended purposes are encoded using the purpose encodings in the *pt_table*. As intended purposes have their implications, purposes are encoded using values from the *aip_code* or the *pip_code* instead of using their own encodings. Also, purposes can be combined by performing bitwise OR operations on the encodings of the purposes. Consider, for example, the intended purpose of a data element is $\langle\{B, C\}, \{G\}\rangle$ with respect to the purpose tree in Fig. 4(i). Then the AIP of the data element is encoded as '0×1B0', which is the result of (0×130 | 0×080), where | is bitwise OR operator. Note that using stored procedures and GUI tools, the management of intended purposes can be easily carried out. We also emphasize that intended purposes should be modified only by trusted users (e.g., privacy officers). This can be easily achieved by authorizing normal users only the *read* privileges for the columns and tables that contain the intended purposes.

With the encoding methods described above, the purpose compliance can be efficiently checked. Recall that an access purpose is compliant with an intended purpose if and only if the access purpose is not prohibited by PIP and it is allowed by AIP. Thus, the purpose compliance check can be done with two bitwise AND operations as follows. Given the

[3] In this approach, any update on the purpose tree can be expensive. However, note that a change on the purpose tree (i.e., the privacy policy) is a very infrequent event. Our approach is thus designed to make storage and performance efficient rather than to optimize updates. Nevertheless, one can easily improve the efficiency of updates by allowing extra nodes in the purpose tree for future expansions.

[4] A different method is to store intended purposes in separate tables. While storing data and intended purpose in separate tables causes additional overhead introduced by join operations, this approach does not require any change to existing data schemas.

```
Comp_Check (Number ap, Number aip, Number pip)
Returns Boolean
1.  if (ap & pip) ≠ 0 then
2.    return False;
3.  else if (ap & aip) = 0 then
4.    return False;
5.  end if;
6.  return True;

Modifying_Query (Query Q)
Returns a modified privacy-preserving query Q'
1.  Let R1, ..., Rn be the relations referenced by Q
2.  Let P be the predicates in WHERE clause of Q
3.  Let a1, ..., am be the attributes referenced in both
    the projection list and P
4.  Let AP be the access purpose encoding of Q
5.
6.  for each Ri where i = 1, ..., n do
7.    if (Ri is relation-based labeling AND
       Comp_Check (AP, Ri.aip, Ri.pip) = False) then
8.      return ILLEGAL-QUERY;
9.    else if Ri is attribute-based labeling then
10.     for each aj which belongs to Ri do
11.       if Comp_Check (AP, aj.aip, aj.pip) = False then
12.         return ILLEGAL-QUERY;
13.       end if;
14.     end for;
15.   else if Ri is tuple-based labeling then
16.     add ' AND Comp_Check (AP, Ri_aip, Ri_pip)' to P ;
17.   else if Ri is element-based labeling then
18.     for each aj which belongs to Ri do
19.       add ' AND Comp_Check (AP, ai_aip, aj_pip)' to P;
20.     end for;
21.   else // Ri is a relation without labeling
22.     do nothing;
23.   end if;
24. end for;
25.
26. return Q with modified P;
```

**Fig. 5** Query modification algorithm

encodings of an access purpose,[5] AIP and PIP, say ap_code, aip_code and pip_code, respectively, the access purpose is compliant with the intended purpose if and only if (ap_code & pip_code) = 0 $\wedge$ (ap_code & aip_code) $\neq$ 0, where & is bitwise AND operator and $\wedge$ is logical AND operator. Method Comp_Check in Fig. 5 illustrates the computation for purpose compliance check.

### 6.2 Access control using query modification

Privacy-preserving access control mechanisms must ensure that a query result contains only the data items that are allowed for the access purpose of the query. In other words, the system must check the intended purpose of each data element accessed by the query and filter out any data element if the access purpose is not compliant with the intended purpose of the data element. In our implementation, this fine-grained access control is achieved using query modification [29]. Note that query modification provides powerful and flexible controls without requiring any alteration in underlying mechanisms and that it is supported in a major commercial DBMS [23,24].

[5] Access purposes are represented using the values in the *code* of the *pt_table*.

Our query modification algorithm is outlined in Fig. 5. Method Comp_Check returns the result of the purpose compliance check for the given purpose labels as described in Sect. 6.1, and method Modifying_Query modifies the given query by attaching additional condition for the purpose compliance check. The complexity of our query modification algorithm is in $O(n)$, where $n$ is the number of attributes accessed by the given query, provided that every relation is labeled with the element-based scheme. Note that method Modifying_Query is invoked only if the access purpose of the query is verified to be acceptable by the validate function, as described in Sect. 3. If the access purpose is not acceptable, then the query is rejected without being processed further.

In Lines 7 and 9 the compliance checks for relations with the relation- or attribute-based labeling schemes are executed statically by the query modification method. On the other hand, the compliance checks for relations with the tuple- or element-based labeling schemes are performed during query processing by the predicates which are added by the query modification algorithm (Lines 15 and 17).

The query modification algorithm checks both the attributes referenced in the projection list and the attributes referenced in predicates (Line 3). As the attributes in the projection list determine what data items will be included in the result relation of a query, it may seem enough to enforce privacy policy based only on the attributes in the projection list. However, the result of a query also depends on the predicates, and not enforcing privacy constraints on the predicates may introduce inference channels. For example, consider the following query: "**SELECT** name **FROM** customer **WHERE** income > 100,000 **FOR** Third-Party", then according to the established privacy policies, *name* can be accessed for the purpose of *Third-Party*, but *income* is prohibited for this purpose. If the privacy constraint is not enforced on the predicates, this query will return a record containing the names of customers whose income is greater than 100,000. This is highly undesirable as this result implicitly conveys information about the customers' income. Note that if the privacy policy is enforced at the predicate level, such inference channels cannot be created.

Notice that the provided algorithm filters out a tuple if any of its elements that are accessed is prohibited with respect to the given access purpose. For instance, consider the following query: "**SELECT** name, phone **FROM** customer **FOR** Marketing". Suppose there is a customer record of which the *name* is allowed for marketing, but the *phone* is prohibited for this purpose, then our algorithm excludes the record from the query result. We note that in the environments where partially incomplete information is acceptable, the query modification algorithm can be easily modified to mask prohibited values with null values using the case expression in SQL.

**Table 7** Query modification examples

| Query | Modified query |
|---|---|
| SELECT name, phone<br>FROM customer<br>FOR Marketing | SELECT name, phone<br>FROM customer<br>WHERE Comp_Check('0x200', name_aip, name_pip)<br>AND Comp_Check('0x200', phone_aip, phone_pip) |
| SELECT name, city<br>FROM customer AS C,<br>address AS A<br>WHERE C.c_id = A.c_id<br>FOR Shipping | SELECT name, city<br>FROM customer AS C, address AS A<br>WHERE C.c_id = A.c_id<br>AND Comp_Check('0x400', addr_aip, addr_pip)<br>AND Comp_Check('0x400', name_aip, name_pip)<br>AND Comp_Check('0x400', A.c_id_aip, A.c_id_pip) |
| SELECT product<br>FROM order<br>WHERE c_id = 1101<br>FOR Profiliing | SELECT product<br>FROM order<br>WHERE c_id = 1101 |

*Example* 5 Table 7 illustrates how queries are modified by our algorithm. Tables 2–6 are used for this example. Note that the purpose encodings of *Marketing* and *Shipping* are assumed to be '0×200' and '0×400', respectively.

## 7 Experimental evaluation

The main goal of our experiments is to investigate performance and storage overheads of our approach. As the relation- and attribute-based labeling schemes do not add significant runtime overheads, we mainly focus on the tuple- and element-based labeling schemes. We consider the impact of the purpose hierarchy and compare the performance overheads of different labeling schemes. We also examine the response times of queries, varying the numbers of attributes accessed. Lastly, we test the scalability of our approach by experimenting with relations of different cardinalities.

### 7.1 Experimental setup

The experiments were performed on a 2.66 GHz Intel machine with 1 GB of memory. The operating system on the machine was Microsoft Windows XP Professional Edition, and Oracle Database 10g Enterprise Edition Release 1 was used as our DBMS.

For the experiment, we generated synthetic datasets, which were simpler versions of the Wisconsin Benchmark [7]. Specifically, each of our datasets consists of two numeric and five string columns, and the data values were generated according to the specification given in [7]. Then we extended each relation by adding intended purpose label columns, according to our labeling schemes described in Sect. 5. In order to compare the overheads of different intended purpose labeling schemes precisely, we set the selectivity of all data elements to 100%, i.e., all intended purpose labels are set to allow access for every access purpose.

After creating all necessary relations, we measured the response times of various queries and the modified versions[6] of those queries. As the main purpose of our experiments was to investigate the overhead introduced by our purpose compliance checks, all of the tested queries were devised to select all the tuples in the target dataset. However, we varied the columns that were accessed by each query. To measure the response time of a query, we measured the time to retrieve the selected tuples into a relation; thus, the reported response times here include the time for inserting the selected tuples, in addition to the time for retrieving the tuples. We ran each query ten times, flushing the buffer cache and the shared pool before each run.

### 7.2 Impact of labeling schemes and purpose hierarchy

Figure 6 shows the response times of queries against relations with various labeling schemes. As shown, the response time increases as the granularity of labeling scheme becomes finer. This increase is indeed expected as more purpose-compliance checks are needed for finer labeling schemes. For the element-based labeling scheme, the number of attributes accessed by queries is also a major factor. As the element-based labeling scheme requires a compliance check for every element a query is accessing, the overhead for compliance checks becomes more significant as the number of accessed attributes increases. However, as the tuple-based labeling scheme requires only one purpose-compliance check for each tuple, the number of accessed attributes does not have any

[6] As we had not implemented the query modification algorithm, each query was modified manually before the experiment. Our future work includes a full implementation of the query modification method in a public domain DBMS.

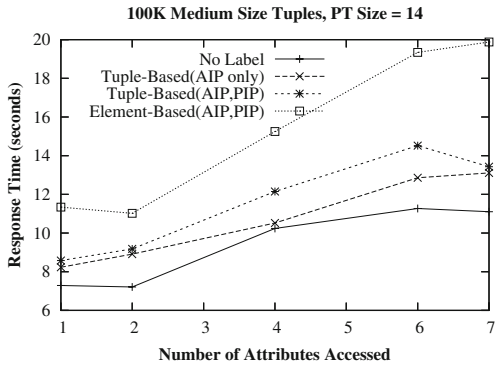



**Fig. 6** Labeling scheme and performance

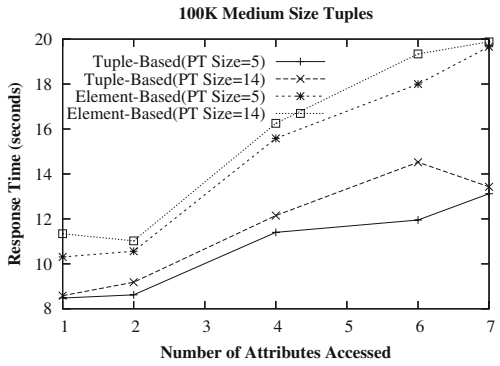



**Fig. 7** Purpose size and performance

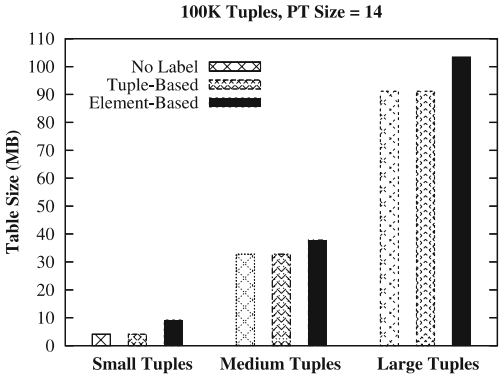



**Fig. 8** Storage and labeling scheme

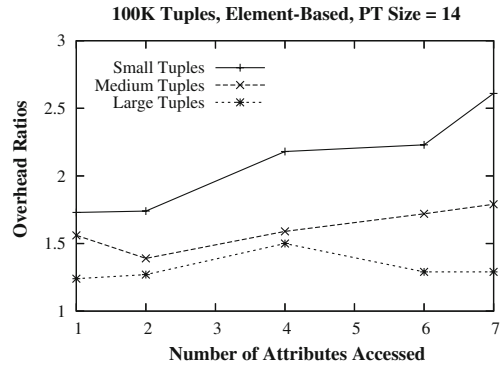



**Fig. 9** Storage and performance

impact on the tuple-based labeling scheme. Figure 6 also shows that the use of both AIP and PIP does not introduce much overhead, compared to the case where only AIP is used. This is a reasonable result as using AIP and PIP requires only one additional bitwise-AND operation for a compliance check.

Figure 7 shows the results of our experiments with two different sizes of purpose trees. The first purpose tree has five nodes with the height of two, and it requires five bits to encode all possible intended purposes. The second purpose tree has 14 nodes with the height of five, requiring 14 bits for all possible intended purpose encodings. As the result shows, the size of purpose tree does not make any substantial difference in either the tuple- or element-based labeling scheme. The reason for this indifference is that bitwise-AND operations are very efficient regardless of the length of encodings.

### 7.3 Storage overhead versus performance overhead

In this section, we consider the storage overhead introduced by our intended purpose labeling schemes. In fact, the storage overheads of labeling schemes can be easily calculated as follows. Let $r_c$ be the cardinality of the relation $\mathcal{R}$, $t_s$ be the size of a tuple and $n_c$ be the number of columns in $\mathcal{R}$. Also, let $p_s$ be the size of the purpose tree, and $\ell_s$ be the size of an intended purpose label; $\ell_s = (p_s/8)$ in bytes. Then the size of the unextended relation is $(r_c \times t_s)$, and assuming the element-base labeling scheme, the size of the extended relation $\mathcal{R}_{\text{ext}}$ becomes $r_c \times (t_s + 2 \times n_c \times \ell_s)$. We remind readers that the element-based labeling scheme extends a relation with $n$ columns to a relation with $(n + 2n)$ columns; $n$ data columns plus $2n$ columns for AIP and PIP. Thus, the storage overhead rate of the element labeling scheme becomes $1 - [\text{Size}(\mathcal{R}_{\text{ext}})/\text{Size}(\mathcal{R})] = (2 \times n_c \times \ell_s)/t_s$. This clearly suggests that if the original relation contains large data elements (which we believe is a common case for databases storing information about individuals), the storage overhead of our labeling scheme becomes negligible. Figure 8 shows the storage overheads of the element-based labeling scheme with relations of three different tuple sizes: small, medium and large. Note that all three relations have the same cardinality of 100K.

The storage overhead has an effect on the performance of modified queries as well. Figure 9 shows performance overhead ratios of modified queries for relations with three different tuples sizes. Here, the overhead ratio is measured as [(response time of the modified query)/(response time of the unmodified query)]. In particular, the table with small-sized

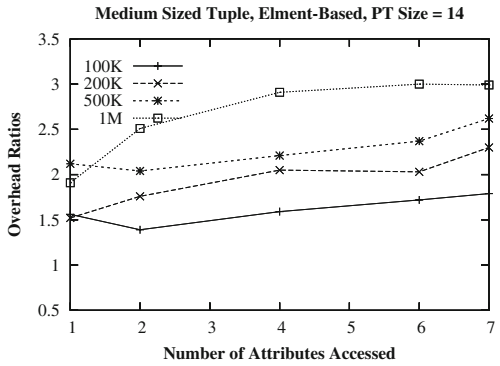



**Fig. 10** Cardinality and performance

tuple is doubled in its size by intended purpose labeling, and the performance overhead on this extended table becomes very large. However, this is an extreme case as typical private data would be much larger in its size than the size of an intended purpose label.

### 7.4 Scalability

As our method filters out prohibited values by performing a purpose-compliance check for each tuple in case of the tuple-based labeling scheme or for each element in case of the element-based labeling scheme, the cardinality of relations can significantly impact performance. For instance, querying a relation with one million tuples will require at least one million compliance checks. Even though our implementation of the compliance check is merely two bitwise-AND operations, the runtime compliance checks can be a heavy burden for large databases. Figure 10 shows the performance overhead rates of the element-based labeling scheme for relations with various cardinalities. As expected, the performance becomes poorer as the cardinality of relations increases. However, this problem can be easily addressed by using function-based indexes [24]. The function-based index allows creating indexes on a function by pre-computing the given function. Thus, we can pre-build a function-based break index for each possible access purpose and use these indexes when queries are executed. Figure 11 displays the results of our experiment with two indexes; a bitmap index and a B+ tree index. Notice that as the compliance check is always evaluated to either true or false (0 or 1), the bitmap index performs slightly better than B+ tree index. Nonetheless, a huge performance improvement is gained when either type of index is used. This shows that using indexes, our approach introduces minimal performance overhead and is highly scalable.

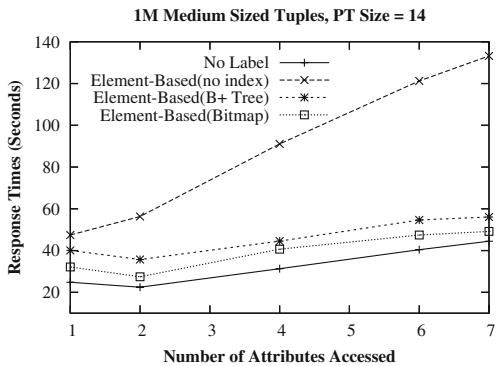



**Fig. 11** Effect of index

## 8 Conclusion and future work

In this article, we proposed an access control model for privacy protection based on the notion of purpose. We discussed a comprehensive definition of purpose and introduced the concepts of intended purpose and access purpose. We proposed an efficient method for determining access purposes, which uses the notions of role attributes and conditional roles. For data labeling, we presented four different labeling schemes, each providing a different granularity. We also discussed various implementation issues and suggested a method based on query modification. Through our experiments, we showed that our method introduces minimal overheads in both storage and performance and is highly scalable.

Our proposed model provides a comprehensive framework for privacy preserving access control systems, but much more work still remains to be done. Our future work includes devising a high level language for purpose-oriented privacy policy which can be used to automatically manage the intended purposes of data. Compatibility issues with P3P will also be investigated. We also plan to extend our model to cope with other elements of privacy such as obligations and complex conditions. In order to achieve this, we will introduce event-based privacy management, which makes use of trigger mechanisms. we will also explore the notion of sticky-policy paradigm [19]. The sticky-policy paradigm requires that the policy under which data have been collected governs the use of these data at all times. This is a challenging problem, but we believe that this is a vital element of privacy protection. We have also observed a need for specialized benchmarks for private data management systems.

To improve our current implementation, we plan to investigate techniques to support our proposed extensions to SQL without requiring actual extensions to the DBMS internals. We also plan to conduct comprehensive usability testing, the result of which will be incorporated in the further improvement of our framework.

**Acknowledgements** This material is based upon work supported by the National Science Foundation under Grant No. 0430274 and the sponsors of CERIAS.

## Appendix: a SQL extensions

In this appendix, we present some examples of SQL extensions that support the purpose management approach presented in this article.

Data definition language (DDL)

1. (***Table Creation***) The purpose labeling specification will be stored in the system catalog and the provided purposes will be used as default values.
   **Create** table-name {
   column-name1 data-type,
   column-name1 data-type,
   . . .
   } [*WithLabeling*]
   where *Labeling* is one of the followings:
   1. EBL(purpose1, purpose2, . . .) : Element-based
   2. TBL(purpose) : Tuple-based
   3. ABL(purpose1, purpose2, . . .) : Attribute-based
   4. RBL(purpose) : Relational-based
2. (***Column Addition***) The table must be element- or attribute-based labeling. If not provided a default purpose (e.g., none-allowed) will be used.
   **Alter Table** table-name
   **Add** column-name data-type [**With** *purpose*]
3. (***Column Removal***) No change is needed.
   **Alter Table** table-name
   **Drop Column** column-name

Data manipulation language (DML)

1. (***Query***) If **For** clause is not provided, the most general purpose (i.e., the root of the purpose tree) will be used.
   **Select** column-names
   **From** table-names
   **Where** column-name = some-value
   [**For** *purpose*]
2. (***Insertion***) The table must be tuple- or element based labeling. If **With** clause is not provided, the default purpose, specified at the creation of table, will be used.
   **Insert into** table-name
   **Values** (v1, v2, . . .)
   [**With** (*purpose*1, *purpose*2,. . .)]
3. (***Deletion***) If **For** clause is not provided, the most general purpose (i.e., the root of the purpose tree) will be used.
   **Delete from** table-name
   **Where** column-name = some-value
   [**For** *purpose*]
4. (***Update***) If **For** clause is not provided, the most general purpose (i.e., the root of the purpose tree) will be used.
   **Update** table-names
   **Set** column-name = new-value
   **Where** column-name = some-value
   [**For** *purpose*]

Purpose management language (PML)

1. (***Purpose Creation***) The root of the purpose tree (e.g., General-purpose) is initially created by the system.
   **Create Purpose** purpose-name
   **Parent** purpose-name
2. (***Purpose Deletion***) If the purpose is not a leaf, the descendants of the purpose will be deleted as well.
   **Delete Purpose** purpose-name
3. (***Intended Purpose View***) If the table is element- or tuple-based labeling, both column-name and value must be provided. For attribute-based labeling, only column-name is required. For relation-based labeling, none is required.
   **View Purpose** table-name
   [column-name] [= value]
4. (***Intended Purpose Update***) If the table is element- or tuple-based labeling, both column-name$_1$ and **Where** clause must be provided. For attribute-based labeling, only column-name$_1$ is required. For relation-based labeling, none is required.
   **Update** table-names
   **Set Purpose** [column-name$_1$ = ] new-purpose
   [**Where** column-name$_2$ = some-value]

## References

1. Agrawal, R., Kiernan, J., Srikant, R., Xu, Y.: Hippocratic databases. In: Proceedings of the 28th International Conference on Very Large Databases (VLDB) (2002)
2. ANSI: American national standard for information technology — role based access control. ANSI INCITS 359–2004 (2004)
3. Ashley, P., Powers, C.S., Schunter, M.: Privacy promises, access control, and privacy management. In: Third International Symposium on Electronic Commerce (2002)
4. Barker, S., Stuckey, P.J.: Flexible access control policy specification with constraint logic programming. ACM Trans. Inf. Syst. Secu. **6**(4), 501–546 (2003)
5. Bell, D.E., LaPadula, L.J.: Secure computer systems: mathematical foundations and model Technical report, MITRE Corporation (1974)
6. Bertino, E., Jajodia, S., Samarati, P.: Database security: research and practice. Inf. Syst. **20**(7), 537–556 (1995)
7. Bitton, D., DeWitt, D.J., Turbyfill, C.: Benchmarking database systems: a systematic approach. In: Ninth International Conference on Very Large Data Bases (1983)

8. Chen, F., Sandhu, R.: Constraints for role-based access control. In: The first ACM Workshop on Role-based access control (1996)
9. Denning, D., Lunt, T., Schell, R., Shockley, W., Heckman, M.: The seaview security model. In: The IEEE Symposium on Research in Security and Privacy (1988)
10. Dong, X., Halevy, A., Madhavan, J., Nemes, E.: Reference reconciliation in complex information spaces. In: ACM International Conference on Management of Data (SIGMOD) (2005)
11. Federal Trade Commision: Children's online privacy protection act of 1998. Available at www.cdt.org/legislation/105th/privacy/coppa.html
12. Federal Trade Commission: Privacy online: fair information practices in the electronic marketplace: a report to congress, May 2000. Available at www.ftc.gov/reports/privacy2000/privacy2000.pdf
13. Fellegi, I.P., Sunter, A.B.: A theory for record linkage. J. Am. Stat. Assoc. (1969)
14. Ferraiolo, D.F., Richard Kuhn, D., Chandramouli, R.: Role-Based Access Control. Artech House (2003)
15. Ferraiolo, D.F., Sandhu, R.S., Gavrila, S., Kuhn, D.R., Chandramouli, R.: Proposed NIST standard for role-based access control. ACM Trans. Inf. Syst. Sec. **4**(3), 224–274 (2001)
16. Goh, C., Baldwin, A.: Towards a more complete model of role. In: The 3rd ACM workshop on Role-based access control. (1998)
17. IBM: The Enterprise Privacy Authorization Language (EPAL). Available at www.zurich.ibm.com/security/enterprise-privacy/epal
18. Jajodia, S., Sandhu, R.: Toward a multilevel secure relational data model. In: ACM International Conference on Management of Data (SIGMOD) pp. 50–59. ACM Press, New York (1991)
19. Karjoth, G., Schunter, M., Waidner, M.: Platform for enterprise privacy practice: Privacy-enabled management of customer data. In: The 2nd Workshop on Privacy Enhancing Technologies (PET 2002) (2002)
20. Kobsa, A.: Personalized hypermedia and international privacy. Communic ACM. **45**(5), 64–67 (2002)
21. Kumar, A., Karnik, N., Chafle, G.: Context sensitivity in role-based access control. ACM SIGOPS Oper. Syst. Rev. **36**(3), 53–66 (2002)
22. LeFevre, K., Agrawal, R., Ercegovac, V., Ramakrishnan, R., Xu, Y., DeWitt, D.: Disclosure in hippocratic databases. In: The 30th International Conference on Very Large Databases (VLDB) (2004)
23. Oracle Corporation: The Virtual Private Database in Oracle9iR2: An Oracle Technical White Paper, January 2002. Available at www.oracle.com.
24. Oracle Corporation: The Oracle Database SQL References, December 2003. Availabe at www.oracle.com.
25. Sandhu, R., Chen, F.: The multilevel relational data model. ACM Trans. Inf. Syst. Secu. **1**(1), 93–132 (1998)
26. Sandhu, R., Ferraiolo, D., Kuhn, R.: The NIST model for role-based access control: towards a unified standard. In: Proceedings of the Fifth ACM Workshop on Role-Based Access Control (RBAC 2000), pp. 47–63 (2000)
27. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. IEEE Comput. **29**(2) 38–47 (1996)
28. Sarawagi, S., Bhamidipaty, A.: Interactive deduplication using active learning. In: ACM International conference on Knowledge discovery and data mining (SIGKDD) (2002)
29. Stonebraker, M., Wong, E.: Access control in a relational data base management system by query modification. In: ACM CSC-ER Proceedings of the 1974 Annual Conference (1974)
30. World Wide Web Consortium (W3C): A P3P Preference Exchange Language 1.0 (APPEL 1.0). Available at www.w3.org/TR/P3P-preferences
31. World Wide Web Consortium (W3C): Platform for Privacy Preferences (P3P). Available at www.w3.org/P3P.