Records Retention in Relational Database Systems: Bridging the gap between laws and enforcement actions

Ahmed A. Ataullah, Frank Wm. Tompa
David R. Cheriton School of Computer Science, University of Waterloo, Ontario, Canada
{aataulla, fwtompa}@uwaterloo.ca

Abstract

Relational databases contain a large corpus of intertwined transaction level facts that can collectively (or independently) represent innumerable business records. However the problem of formally identifying related facts in a database that constitute one or more business records, and then specifying retention policies over those records has largely been unaddressed. In this paper we present our proposed framework for defining business records contained in a database system and then specifying retention policies over them. We also highlight several key issues for both policy makers and database administrators to consider when designing and implementing records retention policies over relationally stored records.

1. Introduction

Limited retention and timely disposal of business records are considered to be critical aspects of modern corporate records management practices. The legal consequences of not preserving business records for minimum specified periods along with the liability associated with over-preservation have sparked great interest in automated tools for records retention. Unfortunately, commercial software solutions for records management solutions have only focused their attention on the preservation and timely destruction of corporate emails and documents.

Relational databases on the other hand are far richer data repositories that documents. A vast number of valuable business records can be created on demand, simply by (re)querying the database at any given point in time. Databases are also often organized such that a single physically stored piece of information can simultaneously be part of several records across various functional areas of a business. Not only does this present a unique challenge for policy makers in indentifying pieces of information in a database that constitute valuable business records but it also leads to possible conflicting policy actions (delete vs. protect) especially when policies from various sources are integrated within the database.

In this paper we discuss the implications of creating, managing and implementing data retention policies within a corporate database from a policy perspective. We address the question of how policy makers and database administrators can together bridge the gap between laws and the actions that must be continuously taken (or disallowed) to maintain a database in a legally compliant state. This problem poses significant challenges not only because the storage structures of database systems are often complex and burdened with dependencies, but also because data retention legislation does not offer a direct mapping between what legislators would consider records and the individual pieces of information contained in corporate database.

There are two primary issues that need to be addressed before we can fully understand the nature of the problem at hand. Foremost is that we need to build an accurate and consistent understanding of what the term "record" can entail in a database system. We believe that purely from an information discovery perspective, any arbitrary query written and executed against a database system by an external auditor or forensic analyst can be used as a record in litigation scenarios. Therefore policy makers and privacy officers must be able to understand the data stored in a database and how it can be manipulated to generate records that may be of concern for an organization. Second policy makers should be thoroughly aware of the implications of actions performed on a database intended to ensure compliance with data retention policies. Database systems offer a wide variety of options for compliance but many may not be enforceable or technically feasible to enforce without significantly degrading the performance of an online

database system serving tens or thousands of clients. Even if actions specified by a policy are viable, the question of whether these actions are actually accomplishing the goals of the legal requirements that they represent still remains to be answered.

In this paper we present a framework that allows legal experts in conjunction with database administrators (DBAs) to identify information in a database that can constitute a record and then enforce retention policies on those records. The technical aspects of this framework (performance limitations, enforceability, conflict detection and related details) have previously been presented in the literature [1]. However the details pertaining to how a set of database level actions can be derived from a given legal policy have not been exposed. In this paper we highlight the need for privacy officers to develop a deeper understanding of relational systems and the unbounded number of records that they can generate. We provide a brief discussion of the implications of the use of a query language as a record identification tool and propose a four step process to develop a consistent and verifiably correct set of data retention policies. We conclude this paper by providing an overview of related work and our ongoing research efforts in the field of privacy policy enforcement within relational database management systems

2. Relational Records

Databases are evolving repositories where not only information is continuously being added and removed, but where the physical storage structure (for example table layout) of the database can also change over time. However what makes databases interesting from a policy perspective is their role as a central corporate information storage system. Policy makers from various functional areas of a business may view overlapping portions of the database as distinct policy relevant records. Therefore retention policy conflicts, where one party may want to destroy information and another stakeholder might want to protect the same can naturally arise in this situation. There may also exist interdependencies among data, by which removal of records may not be possible without compromising the overall integrity of the database.

We believe that regardless of the above mentioned technical challenges, a fundamental requirement for any framework for records management specific to database systems is that it should be able to support all definitions of records as expressed by various policy makers within an organization. In other words if a database is open to legal discovery and forensic analysis, then any information that can be derived from the data contained therein can be considered a record. Therefore we consider a record in relational database system (relational record) as *the results generated by a given data retrieval query at a specific point in time*. Since not all records in a database may be policy-relevant, each business record contained in a database on which a particular retention policy needs to be enforced must be explicitly defined as a query. The elegance and flexibility of this definition lies in the fact that modern query languages, such as SQL, can allow users not only to retrieve single rows of information but can also be used to generate much more complex records containing aggregated information from various parts of the database, thereby making the specification of complex records much easier. However a pre-requisite of this approach is that the user (policy maker) must be somewhat familiar with the particular query language being used by the database system. We now informally present a few examples of records in an SQL database system and discuss how they are very different from other types of records such as documents and emails. We use a very simple database structure where only two tables are used to store information about invoices issued by a hypothetical customer facing company:

```
INVOICE: (INV#, DATE, TAX, CLIENT,
          STATUS,…)
LINEITEM:(INV#, ITM#, DESCR, PRICE,…)
```

A typical record that one would see in real life such as a single printed invoice (say that is uniquely identified by the invoice number 784450) can be defined using the following query:

```
DEFINE RECORD R1 AS
SELECT * FROM
INVOICE  INNER JOIN  LINEITEM
WHERE INV# = “784450”
```

However more complex records can also be created within a database system. For example if we wish to have specific retention requirements for invoices issued to a particular client we can also capture more than one invoice with a single query:

```
DEFINE RECORD R2 AS
SELECT * FROM
INVOICE  INNER JOIN  LINEITEM
WHERE CLIENT = “Susan Jones”
```

In the above example R2 defines a record for policy purposes that identifies the information contained in all

invoices that have been issued to a customer named Susan Jones. Note that a query oriented records management framework allows us to define records that have no real world equivalent. For example the following record (R3) captures specific line items contained across all invoices where the sale of a particular pain medication took place:

```
DEFINE RECORD R3 AS
SELECT * FROM
INVOICE  INNER JOIN    LINEITEM
WHERE DESCR = "Pain medication A"
```

The expressiveness afforded to us by an advanced query language such as SQL can lead us to create hundreds and thousands of records that have no visible and physical equivalent. This ability to derive information dynamically somewhat blurs the boundary between data and a factual record. Consider for example a simple hypothetical fact that a customer, John, was 42 years old in 2010. We can manipulate this fact (simply query it differently) to conclude that John was 41 years old in 2009 and that John was 40 years old in 2008 and so on. Certainly this is a simple contrived example of how a large number of facts can be derived from a single data item. However one can certainly appreciate that with terabytes of information all containing rows representing individual facts, the additional power of querying can automatically lead to the creation/discovery of valuable records that may surpass the size of the initial repository by many orders of magnitude.

The reader should note carefully that our definition of a record in a database system does not specify precisely the contents of a record but rather a meta-description of the record. We can also consider a record (query) to be a method of retrieving, collecting and organization specific information stored in the database at any given point in time. Since the database contents are continuously changing, the notion of temporality is strong associated with relational records. The following record (R4) on our invoicing schema serves as an example of a time sensitive record

```
DEFINE RECORD R4 AS
SELECT * FROM
INVOICE   INNER JOIN   LINEITEM
WHERE datediff_days (today – DATE) <= 5
```

R4 retrieves all the details of invoices created in the past 5 days, and we can use such definitions of records that span temporal regions for policy purposes. Note that even though the definition of this particular record is static, its contents (results from executing the query) change from day to day. Consequently the implications of enforcing retention policies on dynamic records must be carefully examined by policy makers.

Finally we must re-emphasize that although a record in our framework will essentially return with a row of information, these rows may themselves be part of numerous tables and subject to existing integrity constraints. A policy-relevant record may contain rows that comprise aggregate and grouped information such as the sums and averages of numeric fields. With the ability to create a wide variety of records in a database, we must first define what it means to protect and delete information in a record before proceeding further to creation, management and resolution of conflicts within policy.

3. Retention Policies

Our review of data retention policies revealed that there are two basic obligations associated with records retention. First we need to protect records from tampering and/or accidental deletion until minimum specified retention periods have elapsed. The Sarbanes-Oxley act is an example of legislation that mandates minimum protective retention periods for corporate financial records. Many of these records, such as invoices, are derived from information contained in a corporate database. The second requirement is that of destruction of records when they have outlived their useful life. Not only is anonymizing records by partially deleting sensitive information becoming a standard corporate practice but legislation such as Health Insurance Portability and Accountability Act (HIPAA) have recently mandated explicit maximum retention periods for certain records that may be stored in a database system.

In our proposed framework, a record can both be *protected* and *destroyed* with appropriate meaning assigned to both protection and destruciton in the context of ongoing transactions in a database. We define protecting a record as ensuring that its contents are not impacted by user-initiated transactions when a particular retention condition holds true. We can also offer support for more fine grained protection in which a policy maker can selectively allow new row insertions into records (allow the result set generated by the query to only increase in size), but can reject modifications to existing rows in the record. If a user-initiated transaction attempts to violate the protection afforded to a record under a retention policy, then that transaction will be rolled back or aborted. A simple example of a protective retention policy (P1) on record R2 defined earlier is as follows:

DEFINE POLICY P1 ON RECORD R2 AS
PROTECT FROM UPDATE
WHEN STATUS = "PAID"

The above policy ensures that any user initiated transaction that attempts to modify contents of paid invoices issued to Susan Jones is rejected.

In contrast with protective retention policies where only user initiated transactions need to be rejected or allowed, destruction of sensitive records requires that explicit actions must be taken by the database system to ensure removal of sensitive information. In general there is no fixed and prescribed method of deleting relational records, simply because relational records may contain information that is not directly stored (or identifiable) in a database. It may be surprising for the reader that there can exist queries/records for which information generated by the query depends on information *not directly contained in a database*. For example, going back to our invoicing system, we can easily write a query that can give us a list of items that a particular customer has *not* purchased by simply differencing the set of products that are sold by the company and the set of items purchased by a customer. Even though what the database stores is a list of actual purchases made by a customer, the record of the purchases *not* made by the customer is easily derivable and thus may be subject to a retention policy.

From our analysis of legal retention policies we foresee that, for the majority of sensitive records and their applicable retention requirements, the actions required for proper disposal of records will be performed on the records themselves. In other words a straightforward mapping from a record to the physical data it encapsulates will be easy to derive for typical business records. In general such a mapping may not be present and may need to be specified explicitly in the set of actions that need to be taken to enforce a given policy. For more details on such records that are not *directly* stored in databases and how policies can be enforced on them the reader is directed to our prior work [1]. Here we simply point out that that legal experts and policy makers should be aware of this particular aspect of information discovery in database systems. Databases and the wealth of information they contain largely depends on how that information can be manipulated to create valuable facts. However these facts have the potential to be subject to data retention legislation and can become a significant liability in litigation scenarios.

Conceptually a destructive retention policy on a record can be viewed as a condition C on the rows of a record and a particular action A that should be taken when the condition C holds true for any row of record. In other words the disposal of records is done by the invocation of actions specified by the policy maker when proper conditions have been met. To illustrate a destructive retention policy let us consider a destructive retention policy P2 defined on R2

DEFINE POLICY P2 ON RECORD R2 AS
DELETE
WHEN datediff_years (*today - DATE)* >= 7

In P2 the record in question contains all invoices (and related line items) issued to Susan Jones. The condition C in the policy ensures that the action A (deletion) should only be invoked for all invoices that are at least 7 years old. The database system will therefore periodically (or continuously) monitor such invoices and delete them as soon as they are more than 7 years old. The primary reason why we need to explicitly define C and A is that for a particular record, there may be many different ways to "destroy" information contained therein. For example consider P3 which takes a different approach to destroying the same record:

DEFINE POLICY P3 ON RECORD R2 AS
SET CLIENT = "Unknown"
WHEN datediff_years (*today - DATE)* >= 7

Note that the actions specified by P3 are executed when the exact same condition as specified in P2 holds true. However *no physical deletion* of information takes place. Instead the actions overwrite (or anonymize) the client information such that the relevant old/expired invoice is no longer associated with Susan Jones. Both P2 and P3, from a one perspective, accomplish the same overall objectives, that is for all rows in record R2 for which the condition specified in the policy holds true, the specified actions ensure that the rows that violate the given condition are no longer a part of R2. However the question of which policy is more reflective of the legal/operational requirements is subject to further discussion.

4. Creating policies that make sense

We believe that the most significant issue concerning data retention within relational databases is the disconnect between policy makers and database administrators. It is certainly not possible for all policy makers and legal experts to fully understand the complex nature of record keeping and dynamic record generation in a continuously changing database. Similarly legal requirements and policies written in natural language are notorious for being vague, and

seemingly common terms such as "patient records" have no obvious meaning for database administrators that work in the realm of structured and well-defined queries.

The specific problem that we address in this paper is that of converting policy requirements into database queries. All database systems operate under some set of rules for querying and retrieval of information and therefore this layer simply can not be avoided. Similarly it is next to impossible to request system specific clarifications for every legal requirement. Since it is common for organizations to interpret legal requirements as operationally necessary, we believe that the solution lies in both the DBAs and legal experts meeting in the middle and developing acceptable definitions of corporate records in a database. Only then can they move further and develop enforcement actions that guarantee compliance.

We propose a four step process to achieving compliance in a database system. The first step is of course to draw up definitions of records. There has been significant research done in the realm of assistive query writing and query generation wizards that can graphically help non-technical users explore a database are commonly available. With the help of such tools and input from DBAs, organizations can draw up their own set of operational, database-specific record definitions as queries. This process does not need to be centralized and each functional area of the business may have its own legal expert working in conjunction with technical representatives of the corporate database administrator.

In the second stage, specific policy actions will be derived on the records to protect them until certain conditions have been met and/or destroy them when they have outlived their life. In this phase the exact meaning of protecting the record and disposing of information contained therein will be defined and examined in the context of legal requirements. The output will be a set of operational policies (actions and conditions) on the previously defined records. Again this process will be a joint effort between the corporate legal team and the database administrators across various parts of the organization.

In both the first and second phase, it is important that organizations adopt a record discovery approach and the administrators assist in the process. If policy makers attempt to perform a simple mapping of existing physical records onto their relational equivalents, then they are certainly bound to miss a significant number of relational records that may be policy relevant. Specific attention should be given to the use of temporal fields such dates (creation date, date of birth, etc.) where time lapse can bring about new record retention obligations. Similarly numeric fields in tables where aggregate information can be involved may need to be examined carefully. Answering the general question of which information, manipulated in a particular manner, may be relevant for a given legal requirement is undeniably hard. However it can be accomplished via a systematic review process aided by the database administrators.

The third step is perhaps the most crucial: the identification of overlapping records (those records that are produced from the same underlying information in a database) and isolation of conflicts. If a record definition is "too broad" or encompasses a large amount of information, it can be split up into several smaller records for ease of management. In this stage a thorough analysis of record definitions and the actions performed on them as a result of policies will be conducted. Potential conflicts among policies (delete-protect) and conflicts between database integrity constraints and policies will be identified and reported to the administrators before implementation. Since corporate databases are often not restricted to geographic regions or functional areas, it is very likely that a policy implemented by one legal team in one region and a specific area of the organization may conflict with that of a different team. Thus we rely on the database system to detect such possible conflicts. Once detected they will be reported back to the administrators for review.

We believe that as part of this step, conflict resolution will be an ad-hoc process, especially if the conflict occurred between two different legal experts' interpretations of the law. In this case, new definitions of records may need to be drawn up, policy actions may need adjustment or the data layout may need changes within the database system. The source of these conflicts (other than the actual rows that will be affected) is not easy to identify. It may be the case that the laws used to derive policies may be conflicting or only that the specific interpretation of the laws is conflicting. Since preference may not always be given to comply with one legal requirement over the other, the matter is simply reported to the system administrators for review and reassessment.

Also at this stage, unenforceable policies will be brought to the attention of the administrator. We mentioned earlier that databases allow us to create records that can be temporal in nature, for example, R4 was presented as an example earlier where it identified all invoices created less than 5 days ago. For such records protection may simply not be possible or well defined. Let us assume that we had a policy which attempted to protect R4 from any changes and let us also assume that we already have invoices that were

created today. Strictly speaking in five days the invoices created today will automatically no longer be part of R4 simply because of the passage of time. In this context we can not "protect" a record from the passage of time so the protection policy is simply unenforceable. Again, administrators will need to decide what to do with such reported temporal conflicts.

Administrators may also be able to identify performance costs associated with record monitoring at this stage and measure the technical feasibility of enforcing retention policies and how they will impact the overall running of the system. If the performance of a customer-facing database is severely degraded, then administrators will have time to optimize and fine tune the system to mitigate the impact of retention policies once they are actually implemented.

The last step in the process is to implement a fully compatible and enforceable set of policies. Administrators deploy the policies and relevant actions in the database and leave the monitoring of records and enforcement actions to the system. Any attempted violations of retention policies made by user transactions are rejected and can optionally be reported to the administrators/corporate privacy officer for further review and auditing.

5. Related Work

Prior work pertaining to records management has largely not addressed the need for a formal framework for identification of records within relational systems. Several policy definition languages such as E-P3P have been introduced, but none are rich enough to model an adversary that can query the database. Similarly approaches to data retention that are document/object based can not be directly mapped onto relational records because of their dynamic nature.

The problem of systematic records retention has recently been examined in several different contexts within the database community. Lu and Miklau proposed a framework for keeping track of database modifications for auditing purposes via which access to historical tuples in a database can be restricted [2]. Their model is largely focused around tracking and auditing modifications to a relational database. Mitra et al. described a system that ensures that external modifications and tampering with a database are not possible by using write-once media to ensure the integrity of the database is maintained even during system crashes [3]. A broader analysis of forensic implications associated with the use of modern corporate database systems has been done by Stahlberg et al. [4]. They point out that several artifacts such as recovery features embedded in most modern database systems, similar to those that can allow recovery of deleted files, can leave seemingly deleted records available for discovery (through forensic analysis) for a very long time. Their work stresses the need to have adequate control mechanisms that can enable and disable such features on demand.

6. Future Work and Conclusion

Our proposed model for systematic data retention takes an information discovery approach and argues that any external auditor or forensic analyst that has access to a corporate database will be able to query it and generate records. We therefore propose a system that acknowledges that fact and allows us to implement data retention policies on arbitrarily defined queries.

However a significant hurdle in making this approach effective is bridging the gap between legislation and structured queries. We believe that in general, there can be many approaches that can significantly reduce the level of effort required in deriving a set of enforcement actions based on legal requirements for data retention. Automated tools that can parse legal policies to generate a set of requirements at the strategic and operational level in the context of relational records will be an essential requirement in the future.

7. References

[1] Ahmed A. Ataullah, Ashraf Aboulnaga, and Frank Wm. Tompa, “Records retention in relational database systems”, Proceeding of the 17th ACM Conference on Information and Knowledge Management (CIKM), 2008, pp. 873-882.

[2] Wentian Lu, Gerome Miklau, "Auditing a Database under Retention Restrictions", Proceedings of the 25th International Conference on Data Engineering (ICDE), 2009, pp. 42-53.

[3] Soumyadeb Mitra, Marianne Winslett, Richard T. Snodgrass, Shashank Yaduvanshi, Sumedh Ambokar, "Auditing a Database under Retention Restrictions", Proceedings of the 25th International Conference on Data Engineering (ICDE), 2009, pp. 162-173.

[4] Patrick Stahlberg, Gerome Miklau, Brian Neil Levine, "Threats to privacy in the forensic analysis of database systems", Proceedings of the 27th ACM SIGMOD 2007, pp. 91-102.