Stop That Query! The Need for Managing Data Use

Prasang Upadhyaya[1], Nick Anderson[1], Magdalena Balazinska[1], Bill Howe[1], Raghav Kaushik[2], Ravi Ramamurthy[2], and Dan Suciu[1]
[1]University of Washington [2]Microsoft Research

ABSTRACT

When valuable data is exchanged or bought, it is frequently encumbered by restrictions on how it may be used. For example, clinical data must not be used in such a way as to expose the patients' identities. To date, these restrictions are enforced only contractually and compliance is checked only manually, if at all. To meet the needs of this growing set of applications, we present the vision for a Data Use Manager (DUM). The DUM is a component of a database system that enables the declarative specification and enforcement of sophisticated data use policies and provides capabilities for both their online enforcement and offline audit.

1. INTRODUCTION

Data has value. Traditionally, data management research and database tools have ignored the value of the data, except for ensuring its persistence, and have only focused on the value, or cost, of query evaluation. As a result, our community has seen decades of work on query processing and optimization, but limited work on monitoring and controlling how data is used.

The value of data and the need to manage its use occur in several different scenarios. We identify three:

Copyrighted Data. Navteq [18] sells one of the most expensive and well-known databases. Their product is a database of maps used by most commercial GPS navigation systems and many websites, including `maps.bing.com`. To purchase Navteq data, one must contact Navteq, have a face to face meeting, negotiate, sign paperwork, and, quite often, involve lawyers in the process. This interaction is needed in order to reach an agreement about complex restrictions on data use and determine an appropriate price. The restrictions on the data usage are interesting from a data management perspective. For example, one restriction is that maps may only be displayed on a small screen; displaying them on larger screens commands a higher price. Another example is that the buyer is not allowed to join the map data with any other dataset; joining, too, can be done only if one pays a higher price. These restrictions differ from customer to customer, and affect the selling price. Navteq's is not a unique example; companies have been buying and selling data through complex bilateral agreements restricting data usage for many years. Today, we see these usage agreements pop-up even when data is sold online in web-based data markets. For example, on the Windows Azure Marketplace [23], each dataset comes with specific "Publisher Offer Terms".

Private Data. Another example is the case of sharing sensitive data between organizations. The HIPAA Privacy Rule requires HIPAA covered entities (most academic research hospitals and health care systems) to remove or obfuscate 18 Protected Health Information (PHI) data elements from any secondary use of patient datasets so as to reduce the risk of potential reidentification. However, the removal of these elements reduces the overall utility of the data and can lead to "over-sanitizing" the data - potentially rendering entire classes of patients invisible [17, 4]. In this scenario, it would be preferable to leave more information in the data so as to facilitate research but to monitor how the data is being used to avoid privacy violations: *e.g.*, that only aggregate queries are executed on the data or that the data is not joined with any other datasets.

Access Controlled Data. In certain dynamic environments like hospitals, access control policies may be coarsely defined. For instance, an example policy could enable certain personnel to access all patients who have been admitted in an emergency. However, consider a scenario where a celebrity (*e.g.*, Obama) is admitted in an emergency, it is possible that certain authorized personnel may access his health records under conditions that are not authorized for the specific care incidence. Thus, even if access control policies are in place, it is still crucial to track data access.

Vision: Data Use Manager. In all examples above, legitimate users have access to a dataset because their organization bought the data (Ex. 1), otherwise acquired the data (Ex. 2), or produced the data itself (Ex. 3). Users, however, are not allowed to manipulate the data in arbitrary ways either due to dataset utilization terms (Ex. 1) or privacy concerns (Ex. 2 and 3). The challenge is that these scenarios go much beyond the standard access control mechanisms available in existing DBMSs. Instead of restricting access to the data, what is needed is a way to *monitor and restrict the operations that users perform on the data.*

In the above scenarios, users are not malicious and should not be considered adversaries. However, we cannot rely solely on the users to adhere to all constraints for two key reasons: (1) Heterogeneity of data access requirements: The

6th Biennial Conference on Innovative Data Systems Research (CIDR '13) January 6-9, 2013, Asilomar, California, USA.

number of base and derived datasets are growing. If each dataset comes with its own restrictions, it is difficult for users to keep track of what they are and are not allowed to do with the data. (2) Limited range of supported data access modes: Restrictions and the fear of violating agreements can discourage users from exploring the data, which may result in missed opportunities for organizations and the evolving translational science research communities.

To address these challenges, we argue for a new type of data management utility: A *data utilization manager (DUM).* While there are some isolated mechanisms studied in prior work such as access control and auditing, as an infrastructure they are incomplete mechanisms. Indeed, DUMs need to support much richer policies than both access control and privacy mechanisms. They must also support much richer semantics, ranging from online access/deny semantics, to offline probabilistic detection of misuse.

In this paper, we lay the foundations for a research agenda centered around building DUMs: We argue that DUMs ought to support data use policies specified declaratively similar to how access control rules are specified today in SQL (Section 3) and discuss the challenges of articulating their semantics (Section 4). We then discuss the benefits and challenges for both *online* monitoring and enforcement of data utilization policies and post-fact, *offline* usage trail analysis (Section 5). We start by more precisely articulating the problem and core challenges (Section 2).

2. DATA USE MANAGEMENT PROBLEM

As our examples illustrate, data use management is motivated in a wide variety of scenarios. Therefore, instead of letting each application independently manage data use, it makes sense to enhance the DBMS infrastructure with this capability. We envision that the DBMS would incorporate the DUM since the DBMS is already trusted by the data owner; to host the DUM outside, we must trust an additional module of the data management infrastructure.

We call this the *data use management (DUM) problem* and define it more precisely as follows:

- **Centralized, relational engine.** We assume that all data takes the form of a relational database that is stored in a single relational DBMS. We do not discuss alternate architectures in this paper, although the DUM problem extends beyond the centralized and beyond the relational settings.
- **Data utilization policies.** At the heart of the problem is a set of data utilization policies. The database administrator or other key stakeholders specify these policies, which can come from agreements with the data provider, from regulations external to the organization (*e.g.*, HIPAA requirements), or from internal regulations (*e.g.*, scope of data license or need-to-know access control policy). Each policy restricts the operations that users are allowed to perform on the data.
- **Users.** Users execute queries over the data. There are three types of users: ordinary users, curious users, and malicious users. Ordinary users are honest users, but they can be careless in how they use the data. Curious users will go beyond the limits of the policies if the system allows them to do so. However, they will not try to circumvent protection mechanisms if they exist. In contrast, malicious users are willing to go to great extents in order to violate policies. Protection against malicious users thus requires a different type of system than protection against only ordinary and curious users [8]. In many circumstances, overly restrictive access rules are detrimental, as in the case of research access to clinical data [4], which is why the data sharing community has focused work on "honest brokers." Thus, we argue that DUMs need to focus primarily on ordinary and curious users.
- **Data use management problem.** The goal of *data use management* is to (1) enable users to easily specify data utilization policies, (2) observe and record the queries that users execute on the data in order to detect policy violations either before or after they occur, (3) support the capability to prevent policy violations from occurring, and (4) enable post-fact analysis to detect policy violations that occurred in the past.
- **Digital Rights Management.** In this paper, we focus on the problem of monitoring how users operate on data within a DBMS. The problem of attaching policies to data when it leaves the DBMS is an interesting research challenge but we do not discuss it here.

Fuzzy semantics:	Privacy Mechanisms [7]	Intrusion Detection Systems [5]
Precise semantics	Access Control [10, 2]	Auditing Systems [2, 14, 12, 9]
	Online mechanism	Offline mechanism

Table 1: Various data management tools related to Data Usage Management.

The DUM is related to, and generalizes, several existing data management tools and techniques that restrict or control the access to data, shown in Table 1. These systems can be classified along two axes: type of semantics, and time of action. Access controls are ubiquitous mechanisms in database systems and simply restrict access to sensitive data; their semantics is precise (grant/deny), and their action is performed online (at query time). The limitation of these approaches is that they do not handle the case when users are allowed to access individual data items but do not have permission to perform certain operations, such as joins, on these data items.

Privacy mechanisms such as differential privacy [7] also aim to restrict access, but do this in a more fuzzy way, since they restrict access to individual records while allowing access to aggregate queries. These techniques are also insufficient for a DUM because they often protect privacy by limiting data access too much, hence affecting its utility [22].

Newer approaches have migrated the action time from online to offline. For example, auditing systems [2, 14, 12] check that data has been accessed in a compliant way, but do this offline rather than online. In contrast, we envision a DUM as a much more agile tool supporting different management capabilities that include: an online access control mechanism; an online warning mechanism; an offline auditing mechanism; and offline fuzzy detection (*e.g.*, "last week the compliance by customer X was about 85%").

Some systems cross multiple dimensions, for example Hippocratic Databases control access to private data both using an online grant/deny mechanism [16] and through an offline auditing system [1]. These systems support policies that control access to certain data items by certain users, and differ from prior auditing work by specifying a richer class of data-dependent access policy restrictions. But they do not consider policies that selectively grant access to data for certain operations (*e.g.*, aggregates) while denying access to the *same* data for other prohibited operations (*e.g.*, joins).

Existing commercial databases also support a limited form of data use monitoring through triggers [19, 20] by specifying policies to determine when to log events for offline auditing. But, many data use policies would be awkward to express directly as a trigger, *e.g.*, a policy that imposes conditions on the lineage of tuples in the query's result. Thus, we need languages specifically designed for data use monitoring.

Hence no system today is sufficient to serve a DUM mechanism in a DBMS.

3. DECLARATIVE DATA USE POLICIES

The center piece of the DUM framework is a declarative Data Usage Policy, in which the system administrator (or other authorized user) specifies in a declarative way how the data in the RDBMS may be used. The systems shown in Table 1 are concerned with access to individual data items. In contrast, a DUM needs to consider data usage in a much broader sense. Consider the following data use policies examples, which we derived from real scenarios:

1. A query may retrieve at most 200 data items. (For example, if a map is to be displayed only on small screens, then any query must return only a limited number of data items.)
2. No query is allowed to join the data with any other dataset (The Navteq example from Section 1.)
3. Only select-count queries are allowed. Under certain conditions the counts themselves are obfuscated by adding or subtracting a random offset (common federated query approach) [8].
4. Only select-count queries are allowed and each count result must be at least 10. This rule is a common usage agreement in exchanging clinical data [21, 3]. It protects against triangulation attacks when a user needs to compare patient counts at different institutions.
5. No user may issue sequential queries that converge in a way that suggests an attempt to reidentify a particular patient through triangulation of query terms - these conditions are often spelled out in data access policies, but can be circumvented by the curious user [15].
6. No query may access both the location and the hobbies of the same person in a social network dataset. (Prevents unauthorized aggregation of personal data.)

The key research challenge is to develop a data model and query language that are simple yet flexible enough to enable the specification of policies with the same level of sophistication as is done today in hand-written agreements. As the above examples show, the policies refer to several important entities: the data itself; the query's answer; the provenance of each query answer; the log of past queries, their answers, and the provenance of their answers; user and time information. The model and query language must therefore capture these important entities. As a straw-man example, Figure 1 shows a possible declarative specification for the last example policy. As the figure shows, the policy is expressed in SQL and assumes a specific data model with relations storing information about queries, query results, and result provenance. Additionally, the policy is expressed in the form of a constraint. If the check condition returns `true`, the policy is violated. This is one approach to the problem that we are pursuing.

```
P6: CHECK EXISTS
    (SELECT *
    FROM LogQuery x, LogQueryAnswer y,
         LogQueryLineage z1, LogQueryLineage z2
    WHERE x.query_id = y.query_id AND y.answer_id = z1.answer_id
    AND y.answer_id = z2.answer_id
    AND z1.table_name = ‘Location’ AND z2.table_name = ‘Hobbies’)
```

Figure 1: Example of a declarative data-use policy.

4. SEMANTICS

The simplest data use policies in the DUM have deterministic semantics: every policy is a predicate, and if the predicate evaluates to true, there is a policy violation.

However, formally defining a predicate can be challenging. Consider the policy that stipulates that a query may retrieve at most 200 data items. First, even for this intuitive policy, it is non-trivial to define when a data item is "retrieved." An item might be considered as retrieved if it is read off the database during query execution. However, if we intend to relate the data items to the query's output, then if the query filters away an item, it should not be considered as retrieved even after being read; to capture this intent, a data item may count as retrieved only if it contributes to the query output's lineage. Second, if policies refer to data items that are composed of multiple tuples (*e.g.*, a book might be the title along with the set of all authors), existing literature [6, 11] on lineage is inadequate since they only consider lineage in terms of individual records; and it is unclear if such composite data items should be considered used if only a few of their constituent tuples have been used.

Further, in some scenarios, data use violations may only be detected statistically, and hence the policies may additionally require probabilistic semantics. Consider a large social network, which allows applications to query information about its members. One application may be allowed to access the members' `current-location`, but not their `hobbies`; while another application may only access the `hobbies`, but not the `current-location`. If the two applications collude and aggregate their individual data, they can gain information that neither was authorized to. To detect such violations, a DUM may occasionally mine the query logs looking for multiple correlations between the sets of users accessed at about the same time by the two applications: when found, such correlations may suggest a potential violation. Probabilistic semantics, thus, allow a DUM to model policies where some of the operations in a user's workflow are not known to the DUM. This permits a richer class of policies to be specified and probabilistically verified.

Thus, defining correct abstractions and semantics to construct predicates, along with a probabilistic interpretation of those predicates is a crucial step in the design of a DUM.

5. ENFORCING POLICIES

Policy enforcement systems must support both online enforcement as well as post-fact auditing. In an online deployment, the DUM monitors every query and rejects those that violate any policy. For offline deployment, every query is executed and logged (possibly with its output and/or provenance) and the policies are verified offline; any violations may be reported to the data owner. The implementation of such a system, however, raises several important challenges.

In the online setting, data monitoring will incur a performance overhead that the DUM must minimize. Consider the policy that stipulates that queries may retrieve at most 200 data items; enforcing it involves computing each query's output's lineage. Except for policies where each data item is a single record and for select-project-join queries, computing lineage can be an order of magnitude slower than plain query execution [11]. Thus, newer techniques are needed that can exploit additional structure in the policies (and not just the queries) such as the fact that they are boolean or that they only refer to a subset of the query's provenance.

Alternatively, in an offline setting, logging enough information to check policies offline can be expensive during query evaluation while logging too little information can increase auditing time. Since we expect policy violations to be rare, it is difficult to amortize such high overheads.

In fact, the optimal solution might use online checks for some queries and offline checks for the rest; or may use approximations that may yield false positives during online checks, which can subsequently be verified offline. Navigating the online-offline tradeoff is a significant challenge in order to make data monitoring efficient.

Finally, to efficiently enforce policies, we need extensions to the query engine stack. While isolated mechanisms like persisting query results [13] (for offline audits), query hints (to control the number of results), and triggers [19, 20] (to track updates) exist, there is no comprehensive mechanism that can implement the policies from Section 3. Here too, rethinking both the interfaces to the query optimizer (in order to restrict plans based on certain properties of operators derived from the policies) as well as developing new query execution mechanisms (*e.g.*, light-weight schemes to check the policies during query execution) are interesting directions for future work.

6. CONCLUSION

In this paper, we presented the need for a *Data Use Manager (DUM)* as a new component of a DBMS. The DUM enables the declarative specification of sophisticated data use policies and provides capabilities for both their online enforcement and offline audit. We presented the requirements for a DUM and the research challenges associated with satisfying them.

7. ACKNOWLEDGEMENTS

This work is partially supported by the National Science Foundation and Microsoft through NSF CiC grant CCF-1047815 and NSF grant IIS-0915054, additional gifts from Microsoft Research, and Balazinska's Microsoft Research Faculty Fellowship.

8. REFERENCES

[1] R. Agrawal, R. J. B. Jr., C. Faloutsos, J. Kiernan, R. Rantzau, and R. Srikant. Auditing compliance with a hippocratic database. In *VLDB*, pages 516–527, 2004.
[2] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Hippocratic databases. In *VLDB*, pages 143–154, 2002.
[3] N. Anderson, A. Abend, A. Mandel, E. Geraghty, D. Gabriel, R. Wynden, M. Kamerick, K. Anderson, J. Rainwater, and P. Tarczy-Hornoch. Implementation of a deidentified federated data network for population-based cohort discovery. *Journal of the American Medical Informatics Association*, 2011.
[4] N. Anderson and K. Edwards. Building a chain of trust: using policy and practice to enhance trustworthy clinical data discovery and sharing. GTIP '10, pages 15–20, New York, NY, USA, 2010. ACM.
[5] S. Axelsson. Intrusion Detection Systems: A Survey and Taxonomy. Technical Report 99-15, Chalmers University, Mar. 2000.
[6] J. Cheney, L. Chiticariu, and W.-C. Tan. *Provenance in Databases: Why, How, and Where*, volume 1. Foundations and Trends in Databases, 2009.
[7] C. Dwork. A firm foundation for private data analysis. *Commun. ACM*, 54(1):86–95, 2011.
[8] K. Emam El, E. Jonker, L. Arbuckle, and B. Malin. A systematic review of re-identification attacks on health data. *PLoS One*, 6(12), 2011.
[9] D. Fabbri and K. LeFevre. Explanation-based auditing. *PVLDB*, 5(1):1–12, 2011.
[10] E. Ferrari. *Access Control in Data Management Systems*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2010.
[11] B. Glavic and G. Alonso. The perm provenance management system in action. SIGMOD '09, pages 1055–1058, New York, NY, USA, 2009. ACM.
[12] R. Hasan and M. Winslett. Efficient audit-based compliance for relational data retention. In *ASIACCS*, pages 238–248, 2011.
[13] `http://www-01.ibm.com/software/data/db2/linux-unix-windows/time-travel-query.html`.
[14] R. Kaushik and R. Ramamurthy. Efficient auditing for complex sql queries. In *SIGMOD Conference*, pages 697–708, 2011.
[15] C. Kushida, D. Nichols, R. Jadrnicek, R. Miller, J. Walsh, and K. Griffin. Strategies for de-identification and anonymization of electronic health record data for use in multicenter research studies. *Medical Care*, 50:S82–S101, 2012.
[16] K. LeFevre, R. Agrawal, V. Ercegovac, R. Ramakrishnan, Y. Xu, and D. J. DeWitt. Limiting disclosure in hippocratic databases. In *VLDB*, pages 108–119, 2004.
[17] B. Malin. A computational model to protect patient data from location-based re-identification. *Artificial Intelligence in Medicine*, 40(3):223–39, 2007.
[18] `http://www.navteq.com`.
[19] `http://www.oracle.com/technetwork/database/security/index-083815.html`.
[20] `http://wiki.postgresql.org/wiki/Audit_trigger`.
[21] M. SN, W. G, M. M, G. V, C. HC, and C. S. Serving the enterprise and beyond with informatics for integrating biology and the bedside (i2b2). *Journal of the American Medical Informatics Association*, 17(2), 2011.
[22] G. Weber, S. Murphy, A. McMurry, D. MacFadden, D. Nigrin, S. Churchill, and I. Kohane. The shared health research information network (shrine): a prototype federated query tool for clinical data repositories. *Journal of the American Medical Informatics Association*, 16(5):624–630, 2009.
[23] `https://datamarket.azure.com/publishing`.