Information Sciences 298 (2015) 407–424

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Automated conflict resolution in collaborative data sharing systems using community feedbacks

Fayez Khazalah [a], Zaki Malik [a,*], Abdelmounaam Rezgui [b]

[a] Wayne State University, Detroit, MI 48202, USA
[b] New Mexico Tech., Socorro, NM 87801, USA

ARTICLE INFO

Article history:
Received 10 February 2014
Received in revised form 12 September 2014
Accepted 21 November 2014
Available online 5 December 2014

Keywords:
Data sharing
Conflict resolution
Trust
Reputation

ABSTRACT

In collaborative data sharing systems, groups of users usually work on disparate schemas and database instances, and agree to share the related data among them (periodically). Each group can extend, curate, and revise its own database instance in a disconnected mode. At some later point, the group can publish its updates to other groups and get updates of other ones (if any). The reconciliation operation in the CDSS engine is responsible for propagating updates and handling any data disagreements between the different groups. If a conflict is found, any involved updates are rejected temporally and marked as deferred. Deferred updates are not accepted by the reconciliation operation until a user resolves the conflict manually. In this paper, we propose an automated conflict resolution approach that depends on community feedbacks, to handle the conflicts that may arise in collaborative data sharing communities, with potentially disparate schemas and data instances. The experiment results show that extending the CDSS by our proposed approach can resolve such conflicts in an accurate and efficient manner.

1. Introduction

A collaborative data sharing system facilitates users (usually in communities) to work together on a shared data repository to accomplish their (shared) tasks. Users of such a community can add, update, and query the shared repository [17] (please see [37,11,2,45,20] for examples of some collaborative projects). While the shared database evolves over time and users extend it continuously, it may contain inconsistent data, as users may have different beliefs about which information is correct and which is not [18]. While a relational database management system (RDBMS) can be used to manage the shared data, RDMSs lack the ability to handle such conflicting data [17].

In most scientific communities [26,44,21,25,28], there is usually no consensus about the representation, correction, and authoritativeness of the shared data and corresponding sources [25]. For example, in bioinformatics, various sub-communities exist where each focuses on a different aspect of the field (e.g., genes, proteins, diseases, organisms, etc.), and each manages its own schema and database instance. Still these sub-disciplines may have sharing links with their peer communities (e.g., a sharing link between genes and proteins sub-communities). A collaborative data sharing system thus needs to support these communities (and associated links), and provide data publishing, import, and reconciliation support for inconsistent data.

* Corresponding author. Tel.: +1 313 577 4987.
E-mail address: zaki@wayne.edu (Z. Malik).

http://dx.doi.org/10.1016/j.ins.2014.11.029
0020-0255/

Traditional integration systems usually assume a global schema such that autonomous data sources are mapped to this global schema, and data inconsistencies are solved by applying conflict resolution strategies ([36,6,4,35,5,34] are example systems). However, queries are only supported on the global schema and these systems do not support any kind of update exchange. To remedy this shortcoming, peer data management systems [3,22] support disparate schemas, but are not flexible enough to support the propagation of updates between different schemas, and handling data inconsistency issues. In contrast, a collaborative data sharing system (CDSS) [26,44,21,25] allows groups of scientists that agree to share related data among them, to work on disparate schemas and database instances. Each group (or peer) can extend, curate, and revise its own database instance in a disconnected mode. At some later point, the peer may decide to publish the data updates publicly to other peers and/or get the updates from other peers. The reconciliation process in the CDSS engine (that works on top of the DBMS of each participant peer) is responsible for propagating updates and handling the disagreements between different participant peers. It publishes recent local data updates and imports non-local ones since the last reconciliation. The imported updates are filtered based on trust policies and priorities for the current peer. It then applies the non-conflicting and accepted updates on the local database instance of the reconciling peer. For the conflicting updates, it groups them into individual conflicting sets of updates. Each update of a set is assigned a *priority level* according to the trust policies of the reconciling peer. The reconciliation process then chooses from each set, the update with the highest priority to be applied on the local database instance, and rejects the rest. When it finds that many updates have the same highest preference or there is no assigned preferences for the updates in a set, it marks those updates as "deferred". The deferred updates are not processed and not considered in future reconciliations until a user manually resolves the deferred conflicts.

1.1. Problem description

The administrator of each peer in a CDSS is usually responsible for declaring and managing trust policies. While the administrator can be expected to define trust policies for a small number of participant peers, the same is not true for a large number of participants. In addition, assuming that a community of hundreds or thousands of members can authorize a user or a group of users to define trust policies for their community may not be plausible. Moreover, a CDSS does provide a semi-automatic conflict resolution approach by accepting the highest-priority conflicting updates, but it leaves for individual users the responsibility of resolving conflicts for the updates that are deferred. However, the assumption that individual users can decide how to resolve conflicting updates is not strong, as users of the community may have different beliefs and may agree or disagree with each other about which conflicting updates to accept and why (i.e., on which bases). Therefore, the challenge lies in providing a conflict resolution framework that requires minimal or no human intervention.

1.2. Major contributions

In light of the above discussion, we propose a conflict resolution approach that uses community feedbacks to handle the conflicts that may arise in collaborative data sharing communities, with potentially disparate schemas and data instances. The focus is to allow the CDSS engine to utilize the feedbacks for the purpose of handling conflicting updates that are added to the deferred set during the reconciliation process. We list our primary contributions below:

- We define a novel conflict resolution approach that extends the CDSS to automate the resolution of conflicts in the deferred set of a CDSS's reconciling peer. We define a distributed trust mechanism to compute the weight for each conflicting update.
- We provide results for a Java-based implementation of our approach that mimics a community of CDSS peers with disparate schemas and sharing needs.
- We compare our approach with similar techniques to show its applicability for real-world scenarios.

The remainder of the paper is organized as follows. Section 2 presents an overview the related works, followed by the proposed approach for automated conflict resolution in a CDSS (in Section 3). An illustrative example of the proposed approach is introduced in Section 4, which is further used in Section 5 to present an experimental evaluation of the proposed approach. We then conclude the paper in Section 6 and provide brief directions for future work.

2. Related work

In this section, we provide a brief overview of related literature on conflict resolution and trust management in peer-oriented environments. Approaches for the problem of inconsistent data have been described in detail in the context of traditional data integration systems. For instance, [36,6,4,35,5,34] described different approaches to conflict resolution while integrating heterogeneous database sources (see [7] for a comprehensive survey about conflict classifications, strategies, and systems in heterogeneous sources).

Approaches for handling conflicts in community shared databases, based on the concept of multi-versioned database, are described in [17,39,18]. In [17], a BeliefDB system enables users to annotate existing data or even existing annotations, by adding their own beliefs that may agree or disagree with exiting data or annotations. A belief database contains both base