Managing Structured Collections of Community Data

Wolfgang Gatterbauer
University of Washington
gatter@cs.washington.edu

Dan Suciu
University of Washington
suciu@cs.washington.edu

ABSTRACT

Data management is becoming increasingly social. We observe a new form of information in such collaborative scenarios, where users contribute and reuse information, which resides neither in the base data nor in the schema information. This "superimposed structure" derives partly from interaction within the community, and partly from the recombination of existing data. We argue that this triad of data, schema, and higher-order structure requires new data abstractions that – at the same time – must efficiently scale to very large community databases. In addition, data generated by the community exposes four characteristics that make scalability especially difficult: (*i*) *inconsistency*, as different users or applications have or require partially overlapping and contradicting views; (*ii*) *non-monotonicity*, as new information may be able to revoke previous information already built upon; (*iii*) *uncertainty*, as both user intent and rankings are generally uncertain; and (*iv*) *provenance*, as content contributors want to track their data, and "content re-users" evaluate their trust. We show promising scalable solutions to two of these problems, and illustrate the general data management challenges with a seemingly simple example from community e-learning ("ce-learning").

1. A VISION: MASSIVE COMMUNITY E-LEARNING WITH PAIRSPACE

We will argue that management of *collections of community data* requires a new abstraction that does not fit well in the common dichotomy of data and schema information. We illustrate this idea with the vision of a massive online question-answer learning community of users, grouped around a hypothetical tool we refer to as PairSpace. We prefer to keep the overall setup simple. This is a concrete community data management scenario that illustrates the main issues in this paper, while at the same time, seems to have a simple relational implementation. Note that the underlying challenges naturally extend to more complex and general *community content management scenarios*.

5^{th} *Biennial Conference on Innovative Data Systems Research (CIDR '11)* January 9-12, 2011, Asilomar, California, USA.

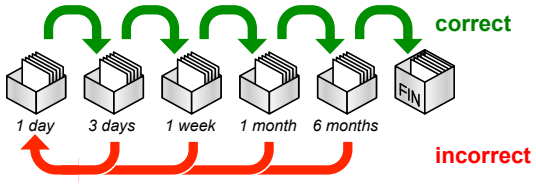

Figure 1: Spaced repetition with flashcard learning: Repetition intervals increase for subsequent boxes.

PairSpace is a huge shared repository of learning nuggets organized into question-answer (Q&A) pairs that combines (*a*) flashcard learning with (*b*) spaced repetition and (*c*) a community built around it. *Flashcards* are sets of cards with a question on one side and an answer on the other. These cards are used as a learning drill to aid memorization of learning material through what is called "active recall[1]:" given a question, one produces the answer. Furthermore, those Q&A pairs are usually grouped into *collections* of a similar nature, i.e. meaningful learning units. Examples are the vocabularies of one lesson in a high-school book, or the standardized questions to pass the US driving license in the State of Washington. Almost any cognitive subject can be translated into such a Q&A format[2].

Spaced repetition is a learning technique with increasing intervals of time between subsequent reviews of learned material. Items to memorize are entered into PairSpace as Q&A pairs (virtual flashcards). When a pair is due to be reviewed, the question is displayed, the user attempts to answer the question, and – after seeing the answer – decides whether he answered it correctly or not. If he succeeds, then the pair gets sent to the next box, if he fails it gets sent back to the first box. Each subsequent box has a longer period of time before pairs are revisited (Fig. 1)[3]. Imagine a daily

[1] In active recall, pieces of information are *actively* retrieved from memory as opposed to *passive review*. See [10] for a recent discussion.

[2] Further examples are: general cultural facts (such as world countries and their capitals), competition results for sports fans (e.g., Who won the 2010 World Cup?), film facts for movie buffs (e.g., Who played William of Baskerville in "The name of the rose" of 1986?), often asked terms during GRE and their synonyms, important paragraphs or cases in law, details on the periodic table in chemistry, multiplication tables in mathematics, names of bones and their location in the human body for medical students, basic formulae in any science, or lists of common abbreviations in computer science (e.g., What does MVD stand for?).

[3] The idea of spaced repetition traces back to the early 1930s, but only became later widely know as Pimsleur's graduated-interval recall [15], or "Leitner system", or "Ebbinghaus Forgetting Curve". While not widely popular in the USA, flashcard learning is hugely popular in Europe with several hundred, mostly offline flashcard programs

morning routine in which a user repeats the learning nuggets that are due that day as suggested by the system.

The third aspect is that those collections of pairs can be shared, re-used, and even re-combined. We envision *one centralized and massive repository* of Q&A pairs for all disciplines, languages and kinds of human knowledge. This is the one central place (with obvious positive externalities) where learners go for repetitive learning needs to find relevant collections of information nuggets, to upload or combine pairs into new collections, and to train regularly. Major value of the stored information lies not just in the individual Q&A pairs (e.g., the translation of English "go" into Spanish "ir" can be easily found in any free online dictionary), but in the *collection of these information nuggets into meaningful units of information* whose mastery together allow the learner to acquire a certain skill level (e.g., passing the knowledge-based driving test). And this value is important to leverage when helping users find the right pairs, collections, or even other users with similar interests.

EXAMPLE 1 (PAIRSPACE SCENARIO). *Alice is learning Spanish. She uplods Q&A pairs of her first lesson. Bob is learning Spanish too and discovers Alice's Spanish 1 lesson in* PAIRSPACE. *His girlfriend is Mexican and has taught him to use* `andar` *instead of* `ir` *for* `go`. *He changes his Q&A pair* (`go`, `ir`) *to* (`go`, `andar`). *Next assume Charlie is searching for basic Spanish lessons. What should the system return to Charlie, and how should it present this result?*

2. CHALLENGES FOR MANAGING COLLECTIONS OF COMMUNITY DATA

The simple scenario of Example 1 already poses several challenges of how to search, return and present the results.

• *What to return?* Should the system return the collection `Spanish 1` of either Alice or Bob? Should it present them as a derivation of each other with Bob's collection as the most recent, or Alice's as the original? Should it return just the intersection of Alice's and Bob's collections as new *derived* collection? Should it present and mark the tuples (`go`, `ir`) and (`go`, `andar`) as possibly *conflicting* pairs? Stated more abstractly, given two or more collections as input, how to inform and return to the user the *structural variation in collections*? How can the system learn to *suggest* new derived collections that fit the purpose of the user?

• *How to bundle and present* the results to the user? Can we take advantage of new "return structures" imposed by collections instead of returning individual pairs in an all-too-familiar list-based fashion (cf. discussion in [3])? If we have several partially overlapping and often complementary or contradicting collections, should we return collections or tuples by majority or by diversity (cf. [20])? Should we cluster these collection into *meta-collections* in the search results, i.e. go one level further in the abstraction?

• *How to search?* How does the user specify the information she is looking for, i.e. what is the appropriate search paradigm for query formulation? What is the appropriate query language that – though possibly hidden from the user – allows to express the user's search needs?

found on the Web. For example, Phase-6 is a German company entirely built around an *offline* flashcard learning software that is used at 3.000 German schools and has supposedly been sold more than 500.000 times (source: `http://www.phase-6.com`). The fact that there are hundreds of software tools available supports the thesis that *one unique, and online* PAIRSPACE would a valuable tool to users, but nobody has yet figured out the perfect solution to bring this to *massive* dimensions (cp. Friendster and LinkedIn before Facebook).

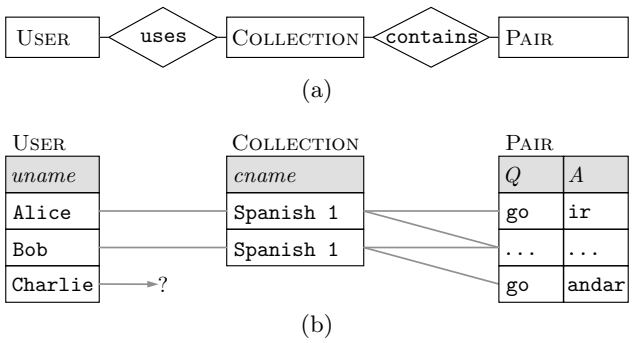

Figure 2: An attempt at a relational encoding of PAIRSPACE: (a) an ER model, and (b) an instance.

• *What to include in ranking?* What are those explicit or implicit features or associations that can be leveraged to learn and return *relevant* information to the user? Obvious candidates are the following: (*a*) *Semantic or syntactic similarity*: How can one address synonymy and polysemy? For example, the expression `bank` can represent "river bank" in English, "bench" in German, and a "financial institution" in both languages. (*b*) *Structure*: What is the generally appropriate way to think about the relative importance between pairs or items and collection of items? (*c*) *Trust or reputation*: What is the appropriate abstraction of trust between users in this scenario? How can different levels of trust be combined, i.e. should they be defined either in a vote-based (democratic, weight-based) manner, or rather a rule-based (strict, preference based) manner. For example, Charlie may specify to trust his school teacher strictly more than any other people. In such preference or rule-based scenarios, the actual value of the weights does not matter, but rather the partial order between preference relations [18]. (*d*) *Provenance*: What kinds of provenance are suggested by this scenario, such as "social provenance" [3], or derivative provenance? Should identical pairs specified by different users be linked to each other? How can one define provenance on collections of items? How to incorporate all those forms of provenance into an appropriate ranking function? How to support querying those combined forms of provenance, e.g. to support *explanatory queries* over this repository?

One fundamental problem is already the question of how to best *store, manipulate, and update* all involved information over time. Figure 2a shows a simple ER model of the relation between users, collections and pairs in our scenario. Figure 2b is a simplified depiction of the scenario of Example 1. If the collection of Alice contains 100 tuples, then storing Bob's *incremental variation* would take $< \frac{2}{100}$th of the space of Alice's original lesson. For the sake of discussion, let's call replicating all pairs as the *explicit representation* (alternatively eager or materialized), and some other representation that stores only the difference as *implicit* (alternatively lazy or virtual). But space is not the major issue here: even if the explicit representation is stored in a compressed form, valuable information about the *relative derivation* or evolution of content is lost. Note, that during querying, value lies not so much in the individual pairs or collections, but rather in the knowledge of how close two or more *collections relate to each other*. In turn, storing only the implicit

information may decrease the access to the actual data considerably. Hence, there is this inherent trade-off between having the data explicit, or the relative derivations explicit. How to update those derivations if content evolves and users add, update, delete or transfer pairs between collections?

Summing up the three main challenges that this seemingly simple scenario of managing three kinds of entities (items, collections, users) in a community scenario raises are: (1) What is the right abstraction for the logical and physical representations of this partly redundant, partly overlapping information, grouped into vastly overlapping bundles? (2) What is the right abstraction of a data manipulation and query language that allows one to reason in terms of collections rather than items? (3) How to evaluate relative importance over the triple concepts of (items, collections, users) in a sound and principled way? In addition, how to reason about (*i*) inconsistency, (*ii*) non-monotonicity, (*iii*) uncertainty, and (*iv*) provenance at the level of collections?

3. WHY EXISTING MODELS AND APPROACHES DON'T SUFFICE

Here we briefly summarize related work that focuses on these challenges but fails short in solving them entirely.

The overall area falls into what is classified as *sharing systems* in [5] where users together build shared structured knowledge bases or a consistent data synthesis. In our scenario, the users do not share the goal of structured knowledge creation, but rather want to find individually fitting collections of Q&A pairs that fit their respective learning needs. Our scenario is clearly different from *data integration* or *data fusion* where the goal is to create *one* unified view on the data [12]. Instead, we want to efficiently manage, find, and compose meaningful collections/bundles/structures of base data that evolve over time. Our challenges are also reminiscent to those of dataspaces [11], where the focus is on incremental ("pay-as-you go") integration. The value of the system increases over time with the number of matches between the data. In our scenario, collections of items have different meanings to different users at different times and need to be managed from day one.

The scenario is also related to the problem of *conflict resolution in community databases* with the goal of automatically assigning each user in the system a unique value to each key [18, 9]. However, in our scenario, content import is "pull" instead of "push," i.e. users actively search for content. In the scenario of searching over Yahoo! answers [1, 2], the goal is to order the set of question-answer pairs according to their relevance to the query. In our scenario, the goal not just to rank just pairs, i.e. user generated content, but rather (or alternatively) *collections* of pairs (which may exist or may be re-combined), or to suggest relevant users.

Our notion of collections is also very reminiscent of *superimposed information* [14], i.e. data that is placed over existing information to help organize and reuse items in these sources. One main difference to superimposed information management is the community aspect: we have different alternative and overlapping collections of base information, and value lies not just in the groupings, but also the *difference between alternative groupings*. The concept of finding and managing associations on top of base data is also related to *inductive databases* [16] and *pattern-base management systems* [4]. Both try to manage rules built upon base data as separate information inside an enhanced DBMS. The difference is that the collections that we are interested in do not have in general an *intensional semantics*. That means they cannot *by themselves* be expressed in a short implicit form, e.g., by a query (cf. [17]). Rather, we are interested in the *incremental and evolving differences* between collections of base data. And we want to leverage this information about "data interference" during the ranking process. Our scenario is also reminiscent of revision control systems (RCS), such as SVN, which manage incremental changes to documents, programs, and other information, and optionally include compression. But while RCSs can store differences efficiently, they do not expose general query facilities to search for meaningful differences, which is an essential ingredient in search in community databases.

4. A NEW HOPE

We propose two complementary approaches to deal with a subset of these challenges.

(1) Rule-based, non-monotone preference relations. An important feature of a community database is providing ways to establish, measure, and show fame/trust/reputation of content and users. This is especially important for a system that needs to provide the users functionality of *revoking* previously stated information ("non-monotone reasoning"). If the user contributions seep deep into the system and other users build upon it, then undoing can be very difficult [5]. Take as an example the spread of wrong information in a social network, where people trust people by a *transitive closure of trust* (people trust people who trust people etc.).

The approach that we promote here is exchanging *vote-based* trust systems (or at least enhancing them) with a *rule-based reputation system* built on *conditional trust*. Conditional trust relations are basically trust mappings at a data-instance level. Trust mapping is a preference-based inference rule or a default statement that a user is willing to accept another user's data value in the absence of their own values. Priorities are further used to specify how to resolve conflicts between data values coming from different trusted users [18]. Conditional trust now allows users to specify selected attributes and thus takes this trust mappings from a schema to a data focus, quite similar to conditional functional dependencies (CFDs) [6] further specifying standard FDs. It is not trivial to reason efficiently in such a rule-based system with the transitive closure of trust, and in the presence of cycles. However, we have shown in very recent work [9] that a unique semantics can be defined that allows to calculate revokes, that means changes in the data instances, efficiently (linear in size of the data and quadratic in the size of the network even in the presence of cycles).

We see two interesting questions with this approach: (*i*) When porting such a rule-based semantics into a richer social context and from the schema-level to data-level, can one still guarantee efficient scalability in both the size of the shared data and the size of the trust relations? (*ii*) Can we raise the level of abstraction for rules further: When treating preference rules as data, can preference-rules on preference-rules still allow an efficient semantics? In recent work [7], we have shown that reasoning in modal logics (arbitrary nesting of annotations on annotations) can be encoded efficiently on top of the relational model. Can we still find efficient encodings of such higher-order rules that can be managed on top of the relational model?

(2) Ranking collections of data in an efficient probabilistic framework. We mentioned before the problem of ranking results relevant to the user, which is also studied in work on incorporating social media into learning-based ranking methods (e.g. [2, 3]). Such ranking methods need to take into account *uncertainty* at several levels: uncertainty about the semantic relevance of the results to the user's query, uncertainty about the user's search context, uncertainty of mapping of keywords to various schema or data information. Furthermore, the ranking method needs to allow for structured queries (e.g. "Find collections on Spanish with the word `go` in one pair and used by trusted users").

A natural way to deal with structured queries over uncertain data is to interpret all uncertainties or weights in the system as probabilistic values in the interval [0,1], and then apply a structured query paradigm that ranks results by their relevance as common in probabilistic databases (PDBs). Recent work on PDBs has shown that the parameters of the ranking functions can be learned from user preferences [13]. Thus, treating the problem of query answering as those of query answering over PDBs allows one to learn the values, then to support complex querying and decision-making over data. The main problem with this approach is its computational complexity: evaluating conjunctive queries over tuple-independent PDBs is well known to be already #P hard, in general. For example, complexity-wise, the above query over the schema from Fig. 2 would correspond to evaluating a Boolean conjunctive chain query $q :- R(x), S(x, y), T(y)$ which is #P hard in our case (many-to-many relations between all entities in Fig. 2a).

The solution that we advocate is our recently introduced technique of *query dissociation* [8]. The basic idea is to slightly change the semantics of probabilistic query evaluation, and to then calculate the ranking score of result tuples with a few materialized views and a single query plan that guarantees efficient evaluation for *every* conjunctive query. In theory, this approach replaces a #P hard problem with a PTIME algorithm. In practice, it allows *existing DBMSs* to evaluate probabilistic conjunctive queries over data instances that have been infeasible for previous approaches.

5. FINAL THOUGHTS

We have pointed to the challenge of organizing and managing *collections of community data* with the example of PairSpace, a fictitious centralized and massive repository for Q&A learning. We envision this shared space as a nucleus that can grow as *the* single repository for general e-learning or even knowledge repositories with more complicated structural knowledge than Q&A pairs (cf. Wikipedia info boxes as nucleus for structured knowledge extraction from the Web). At the same time, solving this structurally simple data management problem will help shape our thoughts of how to approach more complicated structural collections of human knowledge, such as general *community content management systems*, or even those that do not have a naive implementation in the relational model. The most timely such challenge is managing the variance of the human genetic population.

The 1000 Genomes Project (see `http://1000genomes.org`) creates a collection of 1000 human genomes and aims at achieving a complete representation of the structural variation (inserts, deletions, inversions, translocations [19]) in the human genome. Now imagine a massive database that allows to query the human genetic variation of 1 billion genomes, and to correlate these variations with variations in clinical data. This massive data management system clearly derives value from efficient handling of higher-order structure that is imposed on top of the base genetic information (variations between structured collections of base pairs), but also variations in the variations themselves (higher-order structural variations). Effective solutions to biologists for this kind of problem have not yet been found by the data management community.

We strongly believe that investigating the right data model for a massive PairSpace will at the same time help the global learning community, and will lead to better understanding of appropriate abstractions for managing more general massive collections of community data.

Acknowledgement. We like to thank Phil Bernstein and Alexandra Meliou for helpful comments and discussions. This work was supported in part by NSF grant IIS-0915054 (the BeliefDB project: `http://db.cs.washington.edu/beliefDB/`).

6. REFERENCES

[1] E. Agichtein, C. Castillo, D. Donato, A. Gionis, and G. Mishne. Finding high-quality content in social media. In *WSDM*, pp. 183–194, 2008.
[2] E. Agichtein, E. Gabrilovich, and H. Zha. The social future of web search: Modeling, exploiting, and searching collaboratively generated content. *IEEE Data Eng. Bull.*, 32(2):52–61, 2009.
[3] S. Amer-Yahia, L. V. S. Lakshmanan, and C. Yu. Socialscope: Enabling information discovery on social content sites. In *CIDR*, 2009.
[4] I. Bartolini, P. Ciaccia, I. Ntoutsi, M. Patella, and Y. Theodoridis. The PANDA framework for comparing patterns. *Data Knowl. Eng.*, 68(2):244–260, 2009.
[5] A. Doan, R. Ramakrishnan, and A. Halevy. Mass collaboration systems on the world wide web. *CACM*, 2010. (to appear).
[6] W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis. Conditional functional dependencies for capturing data inconsistencies. *ACM Trans. Database Syst.*, 33(2), 2008.
[7] W. Gatterbauer, M. Balazinska, N. Khoussainova, and D. Suciu. Believe it or not: Adding belief annotations to databases. *PVLDB*, 2(1):1–12, 2009. (CoRR abs/0912.5241).
[8] W. Gatterbauer, A. K. Jha, and D. Suciu. Dissociation and propagation for efficient query evaluation over probabilistic databases. In *MUD*, 2010.
[9] W. Gatterbauer and D. Suciu. Data conflict resolution using trust mappings. In *SIGMOD*, pp. 219–230, 2010.
[10] D. Glenn. Close the book. Recall. Write it down. *Chronicle of Higher Education*, 55(34):A1, 2009.
[11] A. Y. Halevy, M. J. Franklin, and D. Maier. Principles of dataspace systems. In *PODS*, pp. 1–9, 2006.
[12] A. Y. Halevy, A. Rajaraman, and J. J. Ordille. Data integration: The teenage years. In *VLDB*, pp. 9–16, 2006.
[13] J. Li, B. Saha, and A. Deshpande. A unified approach to ranking in probabilistic databases. *PVLDB*, 2(1):502–513, 2009.
[14] D. Maier and L. M. L. Delcambre. Superimposed information for the internet. In *WebDB*, pp. 1–9, 1999.
[15] P. Pimsleur. A memory schedule. *The Modern Language Journal*, 51(2):pp. 73–75, 1967.
[16] L. D. Raedt. A perspective on inductive databases. *SIGKDD Explorations*, 4(2):69–77, 2002.
[17] D. Srivastava and Y. Velegrakis. Intensional associations between data and metadata. In *SIGMOD*, pp. 401–412, 2007.
[18] N. E. Taylor and Z. G. Ives. Reconciling while tolerating disagreement in collaborative data sharing. In *SIGMOD*, pp. 13–24, 2006.
[19] E. Tuzun, A. J. Sharp, J. A. Bailey, R. Kaul, V. A. Morrison, L. M. Pertz, E. Haugen, H. Hayden, D. Albertson, D. Pinkel, M. V. Olson, and E. E. Eichler. Fine-scale structural variation of the human genome. *Nat Genet*, 37(7):727–732, 2005 Jul.
[20] C. Yu, L. V. S. Lakshmanan, and S. Amer-Yahia. It takes variety to make a world: diversification in recommender systems. In *EDBT*, pp. 368–378, 2009.