Enabling Change Exploration

Vision Paper

Tobias Bleifuß
Hasso Plattner Institute
tobias.bleifuss@hpi.de

Theodore Johnson
AT&T Labs – Research
johnsont@research.att.com

Dmitri V. Kalashnikov
AT&T Labs – Research
dvk@research.att.com

Felix Naumann*
Hasso Plattner Institute
felix.naumann@hpi.de

Vladislav Shkapenyuk
AT&T Labs – Research
vshkap@research.att.com

Divesh Srivastava
AT&T Labs – Research
divesh@research.att.com

ABSTRACT

Data and metadata suffer many different kinds of change: values are inserted, deleted or updated; entities appear and disappear; properties are added or re-purposed, etc. Explicitly recognizing, exploring, and evaluating such change can alert to changes in data ingestion procedures, can help assess data quality, and can improve the general understanding of the dataset and its behavior over time. We propose a data model-independent framework to formalize such change. Our *change-cube* enables exploration and discovery of such changes to reveal dataset behavior over time.

ACM Reference format:
Tobias Bleifuß, Theodore Johnson, Dmitri V. Kalashnikov, Felix Naumann, Vladislav Shkapenyuk, and Divesh Srivastava. 2017. Enabling Change Exploration. In *Proceedings of ExploreDB'17, Chicago, IL, USA, May 14-19, 2017,* 3 pages.
https://doi.org/http://dx.doi.org/10.1145/3077331.3077340

1 EVER-CHANGING DATABASES

Data change, all the time. This undeniable fact has motivated the development of DBMSs in the first place. While they are good at recording this change, and while much technology has emerged to analyze this data, focus has been on querying and analyzing this data, rather than on exploring and understanding change behavior.

Also: schemata change, quite often. While such metadata-change happens less frequently, schemata are much less stable than what is alluded to in DBMS textbooks, in our experience. And modern "schemaless" DBMSs exacerbate the need to explore changing metadata.

We define the problem of *change exploration* as follows: For a given, dynamic dataset, efficiently capture and summarize changes at value-, aggregate-, and schema-level, and enable users to effectively explore this change in an interactive and graphical fashion. The traditional area of time-series analysis is largely complementary to our vision of change exploration, as explained in Section 2, where we also examine other related research areas.

Beyond the actual changed values, our problem definition includes all traditional investigative questions: *What* was changed? A data value, the name of a property, the validity of a dependency? *Who* induced the change, and *why*, i.e., what is the provenance or underlying process causing the change? *Where* did the change occur, i.e., did it happen in the context of related, simultaneous change events? *When* did the change occur, in particular is there a change pattern over time? *How* was the change effected, i.e., was it an insert, and update or a delete? We are thus interested in a wide variety of these variability aspects over time.

Change happens in all kinds of situations, and consequently we can identify various use-cases for the ability to query and explore such change, including the following.

Profiling. For datasets of unknown content and unknown behavior, change exploration can help understand them, complementing any initial data profiling analysis [1]. For example, assume a column in a dataset does not fit its label and causes confusion. A data change analysis might reveal that over time the column was repurposed but its label remained. A related problem is data lineage or provenance [4], which tries to identify the origin of a piece of data. Our aim goes beyond that: we want to understand the context of change and identify unapparent, implicit links between changes and spot the interesting ones amongst them.

Systemic change. In setups with complex databases, e.g., to manage IP network data, identifying systemic change is important, as it can have impact on how the data should be interpreted. Often, however, analysts only realize that something happened when their results are flawed. Our proposed system can detect and notify about changes in the way a data stream is collected, processed, and/or transported.

Data and schema cleaning. Any additional information about data and metadata dynamics can be exploited to improve data quality. For instance, changes that are rapidly undone may be due to vandalism – information that is unavailable in a static snapshot of the dataset. Or, frequent moves of values between properties point to a poorly designed schema or interface.

In this position paper we propose a general model, the change-cube, and show how it can serve as a basis for a large variety of questions and use-cases to explore changes at data and schema level.

*Research done while on sabbatical leave at AT&T Labs – Research.

ExploreDB'17, May 14-19, 2017, Chicago, IL, USA

ACM ISBN 978-1-4503-4674-0/17/05…$15.00
https://doi.org/http://dx.doi.org/10.1145/3077331.3077340

2 RELATED WORK

Because data change is a fundamental concept of databases, many research areas are related, though none fully cover what we propose. As this short paper cannot afford a full discussion of each, we list the areas and briefly explain their relation to our vision.

Traditional DBMS. With the technology to support transactions, triggers, versioning, logging, incremental view maintenance, etc., database systems include many methods to react to change. The rate of change is an area that is of particular importance to database optimization, in particular how they affect the statistics for query optimization [11]. Our work aims beyond these very specific statistics-based use-cases to a general exploration of data change, and also includes change analysis at schema level.

Temporal and sequence databases. Significant research and development efforts went into the design of databases or database extensions to support temporal or sequence data [10]. For instance, SEQ is a system to support queries not over sets or multisets, but *ordered* collections of records [9]. Many later projects aim at further optimizing such queries and propose specialized query languages.

Our vision can certainly benefit from these previous ideas, which are mostly conceived for relational databases. We plan to determine whether to adapt the techniques and insights to our more generic model of changes or to "outsource" specific analytical queries to such specialized systems.

Temporal profiling. The general idea of adding a temporal dimension to database constraints and other metadata is not new. For instance, temporal association rules [3] are traditional association rules but defined over a (certain) time interval of certain length, during which it has particularly high support (or confidence). Also, the Linked Data community has actively explored methods for the temporal analysis of linked data to understand the processes that populate the data sources and to improve data services [12]. Roussakis et al. [8] distinguish simple and user-defined complex changes in RDF datasets; our approach can discover complex changes.

Data stream mining. A large body of work has proposed analytical methods on data streams [2]. The focus of these methods is mainly on (i) numeric data, (ii) a single dimension/attribute, and (iii) rapidly changing data. The goal is typically a prediction of values, based on past behavior, for instance to predict hardware failure or stock prices, or outlier and pattern detection. In contrast, we want to enable efficient adhoc exploration, we consider data, metadata, and time as equal dimensions, and do not constrain ourselves with the limitations of a streaming environment.

Data and metadata exploration. With more and more relevant data available, the need to interactively explore it has been recognized. For instance, based on profiling results created by the Bellman tool [6], Dasu et al. have explored how data and schema changes in a database can be observed through a limited set of metadata [5]. That work has focussed on the case of only limited access to the database. In contrast, we assume full access to the database and its changes, and are thus able to focus on more fine-grained change exploration. But also in general, we plan to make use of the various recent visualization and interaction frameworks [7], to enable not only static analysis but interactive exploration of the nature of data change.

3 THE CHANGE-CUBE

We choose a generic model to represent changes to a dataset. It includes the following four dimensions to represent *what* changed *where*, *when*, and *how*:

(1) **Time**: A timestamp in the finest available granularity.
(2) **Entity**: The id of an entity that is represented in the dataset. An entity could correspond to a row in a relational database, a node in a graph, a subject of an RDF-triple, etc. Groupings of entities can be modeled separately to represent which rows belong to the same table, which RDF-subjects are of the same class, etc.
(3) **Property**: The property of the entity, corresponding to columns in a table, properties of a graph, predicates of an RDF-triple, etc. Properties can be hierarchically organized, for instance grouped by semantic domain.
(4) **Value**: The new value (or the null-value ($\perp$) to represent a deletion). In principle, values need not be atomic. Values can also be ids of other entities.

Without the time-dimension, the cube represents the traditional data model independent representation of facts as triples. With the time-dimension we can incorporate change:

Definition 3.1. A *change* c is a quadruple of the form

$$\langle \text{timestamp}, \text{id}, \text{property}, \text{value} \rangle \text{ or in brief } \langle \text{t}, \text{id}, \text{p}, \text{v} \rangle$$

Its semantics is: At time t the property p of the entity identified with id was created as or changed to v. A *change-cube* $C = \{c_1, \ldots, c_n\}$ is a set of changes. Among the changes, the combinations of (t, id, p) are unique.

With the uniqueness condition we do not allow multiple simultaneous changes to occur for a specific property of an entity. Without this assumption, a current state of the database would be ambiguous. Further, we assume that id is a stable identifier throughout the lifetime of the representation of the entity. Without this assumption, the notion of "change" would be meaningless.

Implicitly, a value of a change is valid from timestamp until the closest succeeding timestamp of a quadruple with the same id and property but different value. Or it is valid until now, when no succeeding quadruple exists. Note that our model of change also captures changes at schema-level, for instance by the appearance of a new property or other systematic changes to all values of a property, including their deletion. Further, in many real-world situations, additional data is available, such as the id of the person or system that performed the change. Such additions can be managed separately if needed, but are not considered further here.

The change-cube is different from the traditional data cube in three ways: First, instead of affording one dimension for each property/attribute of the data, we gather them all into a single dimension, because we will be asking the same kind of questions for each property or for all properties. In addition, we want to be able to easily add new properties to the model, and explore them. Second, entities of various tables or classes are gathered in a single cube. If needed, they can be hierarchically disambiguated. Third, the domains of all properties are gathered into a single (potentially very large) value-dimension. In this way, a value that appears in multiple locations across properties (and tables) can be recognized as such, for instance to reveal schema changes, such as renaming of attributes.

Populating the change-cube. Nearly all modern databases store logs that allow to derive the quadruples needed to populate the change-cube. In addition, major public-domain datasets contain the information that reflect their change over time as well: Wikipedia, DPLP, Musicbrainz, IMDB, to name just a few. As can be expected, each source represents its changes differently. We have observed the following and expect further variants in the future:

- Timestamped change-records for each field
- Change-records with transaction-ids for each field
- Transaction-logs, with or without timestamps
- Database differences as dump files
- Timestamped snapshots of the entire database

Each of these variants must be consistently transformed into the change-cube representation. In cases where a transaction id is present, we want to use it to populate or refine the timestamp dimension.

Exploring change. The general model of the change-cube allows a wide variety of exploratory analytics, including the use-cases mentioned in the introduction. In this position paper we can only hint at some of them.

As one potential analysis result we propose the novel notion of *volatility* to measure the degree of change, separately for each level of granularity, i.e., for fields, values, entities, properties, for sets of entities (e.g., tables), for entire databases, and for particular timestamps or time intervals. While separate formal definitions are needed due to their different inputs, their goal and intuition are the same: We count and normalize the number of recorded changes (unique timestamps) to the examined item. Thus, we can determine how often a particular field in a database is changed, or for instance compare change frequencies of different properties.

The change of metadata is of particular interest, as it is an under-represented exploration target in research. For instance, we can explore the gradual misusage of schema elements over time. The addition, deletion or renaming of properties/attributes can be tracked – per datatype, per table, or for the entire dataset. Thus, particularly "vulnerable" areas of a schema can be (visually) identified, reminiscent of software-analytics tools that identify frequently changing code areas. Or, gradual changes in the set of valid dependencies can reveal data quality issues.

We envision two styles of analytics: First, one with sets of fixed, domain- and use-case specific queries for known change behaviors and for mining them. Second, we envision an interactive exploration tool allowing users to use a set of operators on the change-cube and thus discover interesting behavior of a dataset at all levels.

4 NEXT STEPS

While this vision paper presented a high-level proposal of the change-cube, we now briefly examine the challenges to create and implement a system[1] to enable change exploration.

Formal underpinnings. While we defined a generic change quadruple and the cube for a set of such changes, we have yet to define measurements taken on the cube, including the mentioned volatility measure and its weights. In addition, we plan to further explore which kinds on change-input our system needs to handle. We were pleasantly surprised by our initial finding of many open sources that publish their changes, but each does so differently.

Use-case specific exploratory analytics. We plan to pursue selected use-cases, one from the open data area and one specific to AT&T, and develop analytical methods to explore the changes in their data. We expect these to be in the form of specific queries that target specific hypotheses. Our learnings from the use-cases shall guide the development of a more general analytical system.

General exploratory analytics. With the ability to query for changes and general change behavior, we want to develop a system that automatically discovers interesting change events and presents these to the user. Also, we want to make use of classification techniques to classify (i) types of change behavior and (ii) data/properties/datasets based in part on their change behavior.

System implementation. To efficiently perform the envisioned types of analyses, we realize based on our experience that we cannot rely on queries directed at some simple physical representation of the change-cube. Rather, we will explore various index structures, and plan to develop highly efficient implementations of exploration primitives.

Interactive exploration. While any automatic analysis can produce interesting results, it is ultimately up to the user to interpret and act upon them. We plan to develop visualizations for data changes and the change-cube, and to allow users to guide the exploration with browsing and querying capabilities, similar to commercial tools like Tableau.

To conclude, we propose a novel view of databases, by regarding their change over time in their data, their metadata, and their constraints. We are confident that this new perspective is a useful addition to the field of data analytics, and we are motivated by the challenges ahead of us.

[1] https:/hpi.de/naumann/projects/data-profiling-and-analytics/dbchex.html

REFERENCES

[1] Ziawasch Abedjan, Lukasz Golab, and Felix Naumann. 2015. Profiling relational data: a survey. *VLDB Journal* 24, 4 (2015), 557–581.

[2] Charu C. Aggarwal. 2007. *Data streams: models and algorithms*. Vol. 31. Springer Science & Business Media.

[3] Juan M. Ale and Gustavo H. Rossi. 2000. An approach to discovering temporal association rules. In *Proc. of SAC*. 294–300.

[4] Peter Buneman, Sanjeev Khanna, and Tan Wang-Chiew. 2001. Why and where: A characterization of data provenance. In *Proc. of the International Conference on Database Theory (ICDT)*. 316–330.

[5] Tamraparni Dasu, Theodore Johnson, and Amit Marathe. 2006. Database Exploration Using Database Dynamics. *IEEE Data Engineering Bulletin* 29, 2 (2006), 43–59.

[6] Tamraparni Dasu, Theodore Johnson, S. Muthukrishnan, and Vladislav Shkapenyuk. 2002. Mining Database Structure; Or, How to Build a Data Quality Browser. In *Proc. of SIGMOD*. 240–251.

[7] Stratos Idreos, Olga Papaemmanouil, and Surajit Chaudhuri. 2015. Overview of Data Exploration Techniques. In *Proc. of SIGMOD*. 277–281.

[8] Yannis Roussakis, Ioannis Chrysakis, Kostas Stefanidis, Giorgos Flouris, and Yannis Stavrakas. 2015. A flexible framework for understanding the dynamics of evolving RDF datasets. In *Proc. of ISWC*. 495–512.

[9] Praveen Seshadri, Miron Livny, and Raghu Ramakrishnan. 1995. SEQ: A Model for Sequence Databases. In *Proc. of ICDE*. 232–239.

[10] Richard T. Snodgrass. 2000. *Developing Time-Oriented Database Applications in SQL*. Morgan Kaufmann.

[11] Michael Stillger, Guy M. Lohman, Volker Markl, and Mokhtar Kandil. 2001. LEO – DB2's LEarning Optimizer. In *Proc. of VLDB*. 19–28.

[12] Jürgen Umbrich, Boris Villazón-Terrazas, and Michael Hausenblas. 2010. Dataset Dynamics Compendium: A Comparative Study. In *Proc. of the International Workshop on Consuming Linked Data (COLD)*.