Discovery and Application of Data Dependencies

Eduardo Henrique Monteiro Pena[1], **Eduardo Cunha de Almeida (orientador)**[1]

[1]Tese de doutorado defendida pela Universidade Federal do Paraná (UFPR)–Brazil

eduardopena@utfpr.edu.br,eduardo@inf.ufpr.br

Abstract. *This work makes contributions that reach central problems in connection with data dependencies. The first problem regards the discovery of dependencies of high expressive power. We introduce an efficient algorithm for the discovery of denial constraints: a type of dependency that has enough expressive power to generalize other important types of dependencies and to express complex business rules. The second problem concerns the application of dependencies for improving data consistency. We present a modification for traditional dependency discovery approaches that enables the dependency discovery algorithms to return reliable results even if they run on data containing some inconsistent records. Also, we present a system for detecting violations of dependencies efficiently. Our extensive experimental evaluation shows that our system is up to three orders-of-magnitude faster than state-of-the-art solutions, especially for larger datasets and massive numbers of dependency violations. The last contribution in this work regards the application of dependencies in query optimization. We present a system for the automatic discovery and selection of functional dependencies. Our experimental evaluation shows that our system selects relevant functional dependencies that help reducing the overall query response time for various types of query workloads.*

Resumo. *Este trabalho faz contribuições que abrangem problemas centrais em relação às dependências de dados. O primeiro problema diz respeito à descoberta de dependências com alto poder de expressividade. Apresentamos um algoritmo eficiente para a descoberta de restrições de negação: um tipo de dependência com poder de expressividade suficiente para generalizar outros tipos importantes de dependências, e expressar regras de negócios complexas. O segundo problema diz respeito à aplicação de dependências para melhoraria de consistência de dados. Apresentamos uma modificação para as abordagens tradicionais de descoberta de dependência que permite que algoritmos de descoberta de dependência retornem resultados confiáveis, mesmo que sejam executados sobre dados contendo alguns registros inconsistentes. Além disso, apresentamos um sistema para detecção eficiente de violações de dependências. Nossa extensa avaliação experimental mostra que nosso sistema é até três ordens de magnitudes mais rápido do que competidores estado-da-arte, especialmente para grandes conjuntos de dados e um grande número de violações de dependência. A última contribuição deste trabalho diz respeito à aplicação de dependências na otimização de consultas. Apresentamos um sistema para a descoberta e seleção automática de dependências funcionais. Nossa avaliação experimental mostra que nosso sistema seleciona dependências funcionais relevantes que ajudam na redução do tempo de resposta para consultas em vários tipos de cargas de trabalho.*

1. Introduction

One of the many essential concepts in relational database management systems and in the relational model regards their capability of enforcement of *constraints* on database objects. There are several applications for constraints: data cleaning, query optimization, query processing, and data integration, just to name a few[Abiteboul et al. 1995, Abedjan et al. 2015]. Constraints are necessary because the relational model, by itself, lacks artifacts that guide the semantic interpretation of tables. Although the names of tables and columns can help us to grasp preliminary meanings of the values in each tuple, they do not specify how these values are related to each other or how we would characterize invalid values. Constraints incorporate such semantics into the relational model because they define semantic properties on a column or group of columns that should be satisfied by relation instances. Unfortunately, the traditional integrity constraint framework–domain constraints, keys, and foreign keys— of most commercial database systems cannot express many critical types of semantic properties. The alternative then is to use *data dependencies* of adequate expressive power to capture such properties.

The initial studies on dependencies started shortly after the proposal of the relational model. Their primary motivation was mainly database design, but nowadays, dependencies are a fundamental part of various data management contexts. Dependencies have a less strict definition than constraints. A dependency is a property on a column or group of columns that apply to particular instances of the database. We can choose a dependency as an integrity constraint. If the database system cannot implement this constraint, then we need to implement it using other means. Different types of dependencies have different levels of expressive power, which means that some of them can restrict inconsistencies that others cannot. The higher the expressive power, the higher the complexity and, thus, the challenge in practical use. That is why the native support for dependencies in database systems is somewhat limited—it is a trade-off between feasibility and expressiveness [Abiteboul et al. 1995].

1.1. Research Challenges

The research questions on types of dependencies of higher expressive power are challenging. Nonetheless, the answers to such questions pursue the development of adequate support for dependencies that can cover a broad range of data inconsistencies. This work makes contributions to some of the central problems in connection with dependencies. We briefly describe these problems in the following.

Discovery of dependencies. Relational database design and maintenance is a complex process that requires, among other tasks, defining sets of constraints. One option is to delegate the task to database designers with adequate expertise in the domain of the application. Although this option may work for small databases and simple types of constraints, it may become infeasible in other scenarios. Database designers with enough expertise might not be conveniently available. Even when experts are around, the manual design of constraints is time-consuming as experts must keep the constraints up-to-date with the semantics of data and application, which is continually evolving. Besides, the higher the expressive power of a dependency language, the higher the complexity in the design of constraints. Finally, the number of possible constraint candidates is usually too large for manual validation, even in small datasets.

The alternative to the manual design of constrains is the automatic discovery of dependencies using data. In a nutshell, the dependency discovery problem is to find the set of dependencies, in a particular language, that holds in a specific table. The problem comes under the umbrella of *data profiling*: the set of activities to gather statistical and structural properties, i.e., *metadata*, about datasets [Abedjan et al. 2015]. The discovery of basic types of dependencies has long been studied [Liu et al. 2012]. In contrast, the discovery of more complex types of dependencies is in the early stages of development, still with a limited number of contributions.

State-of-the-art data cleaning frameworks have used dependencies known as denial constraints (DCs), as they can express complex real-world constraints and generalize other types of dependencies. Each DC φ expresses a relationship between predicates that indicate which combinations of attribute values are inconsistent. A table is inconsistent with a DC φ, if it contains a set of tuples that satisfy all predicates of φ at the same time. For example, given a table citizen with schema $\mathsf{Citizen}(\mathsf{SSN}, \mathsf{Name}, \mathsf{State}, \mathsf{Salary}, \mathsf{Rate})$, we can verify whether all citizens pay a fair amount of taxes with a DC $\varphi_1 \colon \forall \mathsf{t}, \mathsf{t}' \in \mathsf{citizen}, \neg(\mathsf{t.State} = \mathsf{t}'.\mathsf{State} \wedge \mathsf{t.Salary} > \mathsf{t}'.\mathsf{Salary} \wedge \mathsf{t.Rate} < \mathsf{t}'.\mathsf{Rate})$. Any pair of tuples t, t' that satisfy all predicates of φ_1 is a DC violation showing an error in the citizen table. Unfortunately, the discovery of DCs is computationally hard since it regards a search space that is bigger than the search space seen in the discovery of simpler dependencies.

Application of the discovered dependencies. The number of discovered dependencies radically increases with the number of columns in the dataset. This number may increase drastically as the number of columns goes up, e.g., in the region of millions for datasets with hundreds of columns and thousands of records [Papenbrock et al. 2015]. The main problem is that selecting which of the dependencies are most relevant for a given task (e,g, data cleaning) is left for human analysis. It is particularly challenging to understand the relationships among hundreds, or even thousands, of dependencies spread across multiple relations. Interestingness measures have been proposed to score data dependencies [Pena et al. 2019]. These measures are primarily based on the statistical properties of the data and have shown good potential to filter dependencies for tasks such as data cleaning and query optimization [Pena et al. 2018]. As observed in [Kimura et al. 2009], however, data dependencies should be exploited with caution because they may impose additional performance penalties as their number increase. Thus, the goal is to use different strategies to recommend sets of dependencies that offer the best trade-off between a reduced number of dependencies and best gains for a particular dependency use-case.

Detection of dependency violations. Data inconsistencies emerge as violations of the dependencies defined for the database. Knowing to which extend inconsistencies permeates a database is the first step towards producing better-quality query answers; therefore, the detection of dependency violations is vital. In data cleaning pipelines, nothing can be done before the detection step. Even if fixing inconsistencies is not possible, users surely need to be aware of the inconsistencies so they can avoid poor decision-making.

The most straightforward way to detect a dependency violation is to enumerate the necessary combination of tuples, and then check whether each combination complies with the dependency or not. Of course, this approach is impractical for large datasets since it has a quadratic time complexity in the number of tuples. An alternative to the naive ap-

proach is to translate dependencies into SQL queries and then ask a database management system to find the violations. Although the use of database systems is practical, it has two critical performance drawbacks. The performance varies significantly from system to system, and, worst yet, it is usually not robust against different types of dependencies. For the same dataset, a database system may perform well for a given dependency but perform poorly for another.

Most of the recently presented data cleaning systems use database systems to detect violations of data dependencies. Still, their experimental evaluations are quite limited, as they explore mostly simple dependencies (e.g., functional dependencies) and small datasets [Rekatsinas et al. 2017]. In many real-world scenarios, however, data cleaning (and other data management tasks) has to deal with large datasets and complex dependencies such as denial constraints. Thus, there is a need for efficient techniques to detect violations of dependencies of various types.

2. Summary of contributions

The research on data dependencies is vibrant, but at the same time, challenging. Contributions on the field have numerous applications in various data management aspects. The contributions of this work cover four primary dimensions, summarized as follows.

2.1. A novel algorithm for the discovery of denial constraints [Pena et al. 2019].

The alternative to designing denial constraints by hand is automatically discovering denial constraints from data. Unfortunately, this alternative is computationally expensive due to the vast search space derived from the number of predicates that can form denial constraints. To tackle this challenging task, we present a novel algorithm, DCFINDER. It combines data structures called position list indexes, bitwise operations, and optimizations based on predicate selectivity to validate denial constraint candidates efficiently. Because the available data often contain errors, the design of DCFINDER algorithm focuses on the discovery of relaxed denial constraints. Our experimental evaluation uses real and synthetic datasets and shows that DCFINDER outperforms previous existing algorithms for the discovery of relaxed denial constraints.

2.2. A novel technique to focus the dependency discovery in denial constraints useful for data cleaning [Pena and de Almeida 2019].

In the traditional approach to the discovery of dependencies, the results are as reliable as the data used to produce them. Having problematic data is often involuntary; thus, the discovery should be able to accommodate potential data errors. Besides, the number of discovered results grows exponentially with the number of columns in the table. Even if we discover dependencies from correct data, many results may hold only by chance, i.e., they are spurious. We propose a method that uses statistical evidence of the tuples of a dataset to focus the discovery of denial constraints. Our method sets DCFINDER so that it can find denial constraints appropriate for data cleaning, even if the dataset contains errors. Our experiments with real data show that the identified denial constraints point, with high precision and recall, to inconsistencies in the input data.

2.3. A novel system to detect violations of denial constraints [Pena et al. 2020].

Dependencies and their violations can reveal errors in data. Several data cleaning systems use database systems to detect violations of data dependencies. While this approach is

efficient for some kinds of data dependencies (e.g., key dependencies), it is likely to fall short of satisfactory performance for more complex ones, such as some forms of denial constraints. We propose a novel system to detect violations of denial constraints efficiently. We describe its execution model, which operates on compressed blocks of tuples at-a-time, and we present various algorithms that take advantage of the predicate form in denial constraints to provide efficient code patterns. Our experimental evaluation includes comparisons with different approaches; real-world and synthetic data; and various kinds of denial constraints. It shows that our system is up to three orders-of-magnitude faster than the other solutions, especially for datasets with a large number of tuples and denial constraints that identify a large number of violations.

2.4. Novel techniques to detect functional dependencies appropriate for query optimization [Pena et al. 2018].

We present a system for query optimization based on automatic discovery of data dependencies. By formulating query transformations, it can revise the number of processed rows, with a direct impact on performance. The goal is to optimize query execution in cases where the database is denormalized or have lost dependencies in the design. We rely on the automatic discovery of dependencies, but to avoid optimizing for spurious dependencies, we focus on dependencies matching the current queries in the pipe (i.e., the *workload*). Initially, we use a state-of-the-art algorithm to discover the set of functional dependencies holding in the datasets. Then, our *focused dependency selector* uses the available workload information to choose exemplars from the set of the discovered functional dependencies that are appropriate for query optimization. That eliminates any manual interaction. The selected dependencies exhibit statistical properties that resemble those of the initial set of dependencies; therefore, they serve as a semantical summary of the dependencies. We use well-known techniques for query optimization with the selected dependencies. In the best-case scenario of our experimental evaluation, our system can reduce query response time by more than one order of magnitude using join elimination for a real-world database.

3. Relevance of the research

In this work, we contributed to the development of algorithms and systems for supporting data scientists and IT-professionals in their quest for metadata use. In particular, we focused on the discovery and application of data dependencies because of their critical role in data management. Our contributions span publications presented in several high-quality venues [Pena et al. 2018, Pena and de Almeida 2018, Pena 2018, Pena et al. 2019, Pena and de Almeida 2019, Pena et al. 2020]. Some of these contributions are readily available since we integrated some of our prototypes into an open data profiling platform called Metanome [1].

The development of this work allowed us to leverage collaborations with well-known research groups from Europe, namely, the Interdisciplinary Centre for Security, Reliability and Trust (SNT) at the University of Luxembourg, and the Hasso Plattner Institute at the University of Potsdam-Germany. Also, we contributed to work that is orthogonal to our thesis, which also appears as a high-quality publication [Santore et al. 2020].

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

References

Abedjan, Z., Golab, L., and Naumann, F. (2015). Profiling relational data: A survey. *The VLDB Journal*, 24(4):557–581.

Abiteboul, S., Hull, R., and Vianu, V. (1995). *Foundations of Databases*. Addison-Wesley.

Kimura, H., Huo, G., Rasin, A., Madden, S., and Zdonik, S. B. (2009). Correlation maps: A compressed access method for exploiting soft functional dependencies. *Proc. VLDB Endow.*, 2(1):1222–1233.

Liu, J., Li, J., Liu, C., and Chen, Y. (2012). Discover dependencies from data - a review. *IEEE TKDE*, 24(2):251–264.

Papenbrock, T., Ehrlich, J., Marten, J., Neubert, T., Rudolph, J.-P., Schönberg, M., Zwiener, J., and Naumann, F. (2015). Functional dependency discovery: An experimental evaluation of seven algorithms. *PVLDB.*, 8(10):1082–1093.

Pena, E. H. M. (2018). Workload-aware discovery of integrity constraints for data cleaning. In *VLDB 2018 - PhD Workshop*, volume 2175.

Pena, E. H. M. and de Almeida, E. C. (2018). Bfastdc: A bitwise algorithm for mining denial constraints. In *Database and Expert Systems Applications (DEXA)*, pages 53–68, Cham. Springer International Publishing.

Pena, E. H. M. and de Almeida, E. C. (2019). Short paper: Descoberta automática de restrições de negação confiáveis. In *XXXIV Simpósio Brasileiro de Banco de Dados, SBBD 2019, Fortaleza, CE, Brazil, October 7-10, 2019*, pages 187–192. SBC.

Pena, E. H. M., de Almeida, E. C., and Naumann, F. (2019). Discovery of approximate (and exact) denial constraints. *Proc. VLDB Endow.*, 13(3):266–278.

Pena, E. H. M., Falk, E., Meira, J. A., and de Almeida, E. C. (2018). Mind your dependencies for semantic query optimization. *JIDM*, 9(1):3–19.

Pena, E. H. M., Lucas Filho, E. R., de Almeida, E. C., and Naumann, F. (2020). Efficient detection of data dependency violations. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM)*, page 1235–1244.

Rekatsinas, T., Chu, X., Ilyas, I. F., and Ré, C. (2017). Holoclean: Holistic data repairs with probabilistic inference. *PVLDB Endow.*, 10(11):1190–1201.

Santore, F., de Almeida, E. C., Bonat, W. H., Pena, E. H. M., and de Oliveira, L. E. S. (2020). A framework for analyzing the impact of missing data in predictive models. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 2209–2212.