AN ABSTRACT OF THE DISSERTATION OF

Parisa Ataei for the degree of Doctor of Philosophy in Computer Science presented on June 20, 2021.

Title: Theory and Implementation of a Variational Database Management System

Abstract approved: _____

Eric Walkingshaw

In this thesis I present the variational database management system, a formal framework and its implementation for representing variation in relational databases and managing variational information needs. A variational database is intended to support any kind of variation in a database. Specific kinds of variation in databases have already been studied and are well-supported, for example, schema evolution systems address the variation of a database's schema over time and data integration systems address variation caused by accessing data from multiple data sources simultaneously. However, many other kinds of variation in databases arise in practice, and different kinds of variation often interact, but these scenarios are not well-supported by the existing work. For example, neither the schema evolution systems nor the database integration systems can address variation that arises when data sources combined in one database evolve over time.

This thesis collects a large amount of work: It defines the variational database framework and the syntax and [specific kind of] semantics of the variational relational algebra, a query language for variational databases. It also defines the requirements of a generic variational database framework that makes the framework expressive enough to encode any kind of variation in databases. Additionally, it [shows/proves] that the introduced framework satisfies all these needs. It presents two use cases of the variational database framework that are based on existing data sets and scenarios that are partially supported by existing techniques. It presents the variational database management system which is the implementation of variational databases and variational relational algebra as an abstract layer written in Haskell on top of a traditional RDBMS. It also presents several

theoretical results related to the framework and query language, such as syntax-based equivalence rules that preserve the semantics of a query, a type system for ensuring that a variational query is well formed with respect to the underlying variational schema, and a confluence property of the variational relational algebra type system and semantics with respect to the relational algebra type system and semantics.

Theory and Implementation of a Variational Database Management System

by

Parisa Ataei

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented June 20, 2021
Commencement June 2021

Doctor of Philosophy dissertation of Parisa Ataei presented on June 20, 2021.

APPROVED:

Major Professor, representing Computer Science

Director of the School of Electrical Engineering and Computer Science

---

Dean of the Graduate School

I understand that my dissertation will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my dissertation to any reader upon request.

Parisa Ataei, Author

ACKNOWLEDGEMENTS

[Eric. Committee. jeff. abu. parents. friends.]

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF ALGORITHMS

Algorithm Page

Chapter 1: Introduction

Managing variation in databases is a perennial problem in database literature and appears in different forms and contexts [19, 3, 7, 10, 4] and it is inevitable [18]. Variation in databases mainly arises when multiple database instances conceptually represent the same database, but, differ in their schema, content, or constraints. Specific kinds of variation in databases have been addressed by context-specific solutions, such as schema evolution [13, 6, 2, 17, 14], data integration [9], and database versioning [5, 12]. However, there are no generic solution that addresses any kind of variation in databases. We motivate the need for a generic solution to variation in databases in Section 1.1.

The major contribution of this thesis is the *variational database* framework, a generic relational database framework that explicitly accounts for database changes (variation), and *variational relational algebra*, a data manipulation language for our framework that allows for information extraction from a variational database. The variational framework is generic because it can encode any kind of variation in databases. Additionally and more importantly, it is designed such that it satisfies any information need that a user may have in a variational database scenario. Based on information needs in a variational database scenario, we define requirements of a variational database framework in Section 1.1 and throughout the thesis, we show that our framework satisfies these requirements.

In addition to a formal description of the variational database framework and variational relational algebra and some theoretical results, this thesis distributes and presents two variational data sets (including both a variational database and a set of queries) as well as a *variational database management system* that implements the variational database framework as an abstraction layer on top of a traditional relational database management system in Haskell. Section 1.2 enumerates the specific contributions in the contest of an outline of the structure of the rest of the thesis.

1.1 Motivation and Impact

[include a more elaborate version of vldb intro with maybe parts of vamos. also include

the requirements here and elaborate more. you can also refer to sections that address each requirement. finally introduce the mot ex as a concrete running example throughout the thesis. also industry ex as an example that came up in conversation with industry contact.]

Managing variation in databases is a perennial problem in database literature and appears in different forms and contexts [19, 3, 7, 10, 4]. Variation in databases mainly arises when multiple database instances conceptually represent the same database, but, differ in their schema, content, or constraints. Existing work on variation in databases addresses specific kinds of variation in a context and proposes solutions specific to the context of the variation, such as schema evolution [13, 6, 2, 17, 14], data integration [9], and database versioning [5, 12].

Unfortunately, some of these tools do not address all their user's needs. Furthermore, they are all *variation-unaware*, i.e., they dismiss that they are addressing a specific kind of a bigger problem. Thus, not only they cannot address a new kind of database variataion but they also cannot address the interaction of different kinds of database variation since they assume that each kind of variation is *isolated* from another kinds. This costs database researchers to develop a new system for every individual kind of data variation and forces developers and DBAs to use manually unsafe workaround.

For example, schema evolution is a kind of schematic variation in databases over time that is well-supported [13, 6, 2, 17, 14]. Changes applied to the schema over time are *variation* in the database and every time the database evolves, a new *variant* is generated. Current solutions addressing schema evolution dismiss that it is a kind of variation, thus, they *simulate* the effect of variation by relying on temporal nature of schema evolution and using timestamps [13, 6, 2, 17] or keeping an external file of timeline history of changes applied to the database [14].

Unlike schema evolution, some kinds of database variation are partially supported. For example, while developing software for multiple clients simultaneously, an approach called software product line (SPL) [8], a different database schema is required for each client due to client's different business requirements and environments [16]. SPL researchers have developed encodings of data models that allow for arbitrary variation by annotating different elements of the model with features from the SPL [16, 15, 1]. Thus, they can generate a database schema *variant* for each software *variant* requested by a client and generated by the SPL. However, these solutions address *only* variation in the

data model but do not extend to the level of the data or queries. The lack of variation support in queries leads to unsafe techniques such as encoding different variants of query through string munging, while the lack of variation support in data precludes testing with multiple variants of a database at once.

The situation exacerbates even more when two kinds of variation interact and create a new kind of database variation: the evolution of the database used in development of software by an SPL. This is due to the evolution of software and its artifacts; an inevitable phenomena [11]. This is where even previous context-specific solutions like schema evolution tools fail. We motivate this case through an example in Section 1.1.

As we have shown, variation in databases is abundant and inexorable [18]; impacts DBAs, data analysts, and developers significantly [11]; and appears in different contexts. Current methods are all extremely tailored to a specific context. Consequently, they fail to address the interaction of their specific kind of database variation with other kinds. Hence, the challenge becomes defining a variation-aware database that can can model different kinds of variation in different contexts such that it satisfies different specialists' needs at different stages, e.g., development, information extraction, deployment, and testing.

1.1.1 Motivating Example

[combination of instances, behaviours, and dimensions]

1.1.2 New Instance of Data Variation in Industry

[revise base on what you've explained so far in thesis.]

New variational scenarios could appear, either from combination of other scenarios or even a new scenario could reveal itself. For example, the following is a scenario we recently discussed with an industry contact: A software company develops software for different networking companies and analyzes data from its clients to advise them accordingly. The company records information from each of its clients' networks in databases customized to the particular hardware, operating systems, etc. that each client uses. The company analysts need to query information from all clients who agreed to share their information, but the same information need will be represented differently for each

client. This problem is essentially a combination of the SPL variation problem (the company develops and maintains many databases that vary in structure and content) and the data integration problem (querying over many databases that vary in structure and content). However, neither the existing solutions from the SPL community nor database integration address both sides of the problem. Currently the company manually maintains variant schemas and queries, but this does not take advantage of sharing and is a major maintenance challenge. With a database encoding that supports explicit variation in schemas, content, and queries, the company could maintain a single variational database that can be configured for each client, import shared data into a VDB, and write v-queries over the VDB to analyze the data, significantly reducing redundancy across clients.

1.2 Contributions and Outline of this Thesis

[contribution]

Chapter 2: Background

[background]

2.1 Types

[types]

2.2 Relational Databases

[relational database]

2.3 Relational Algebra

[relational algebra]

2.4 Variational Set

[vset]

2.4.1 Variational Set Configuration

[vset configuration.]

2.5 The Formula Choice Calculus

[formula choice calculus]

Chapter 3: The Variational Database Framework

[needs. must have configuration.]

3.1 Variational Needs in a Relational Database

[needs and examples of them.]

3.2 Variation Space in a Variational Database Framework

[fexp. evaluation.]

3.3 Variational Schema

[vsch]

3.3.1 Variational Schema Configuration

[vsch configuration.]

3.4 Variational Table

[vtab]

3.4.1 Variational Table Configuration

[vtab configuration]

3.5 Variational Database

[vdb]

3.5.1 Variational Database Configuration

[vdb configuration]

3.6 Properties of a Variational Database Framework

[well-formed vdb properties.context-specific properties.]
[show that they hold for vdb.]

Chapter 4: The Variational Query Language

[vql]

4.1 Variational Relational Algebra

[vra]

4.1.1 VRA Configuration

[vra configuration]

4.1.2 VRA Semantics

[vra semantics]

4.1.3 VRA Type System

[type sys]

4.1.4 VRA Variation-Minimization Rules

[rules]

4.2 Variational Query Language Properties

[prop. show for vra.]

Chapter 5: Variational Database Management System (VDBMS)

[vdbms]

5.1 Implemented Approaches

[apps]

5.2 Experiments

[exp.]

Chapter 6: Related Work

[related work! have to work on this!]

6.1 Instances of Variation in Databases

[schema evolution. database versioning. data integration. data provenance.]

6.2 Instances of Database Variation Resulted from Software Development

[SPL. data model. query.]

6.3 Variational Research

[blah]

Chapter 7: Conclusion

[conclusion]

Bibliography

[1] Lamia Abo Zaid and Olga De Troyer. Towards modeling data variability in software product lines. In Terry Halpin, Selmin Nurcan, John Krogstie, Pnina Soffer, Erik Proper, Rainer Schmidt, and Ilia Bider, editors, *Enterprise, Business-Process and Information Systems Modeling*, pages 453–467, Berlin, Heidelberg, 2011. Springer.

[2] Gad Ariav. Temporally oriented data definitions: Managing schema evolution in temporally oriented databases. *Data & Knowledge Engineering*, 6(6):451 – 467, 1991.

[3] Parisa Ataei, Qiaoran Li, and Eric Walkingshaw. Should variation be encoded explicitly in databases? In *15th International Working Conference on Variability Modelling of Software-Intensive Systems*, VaMoS'21, New York, NY, USA, 2021. Association for Computing Machinery.

[4] Anant P. Bhardwaj, Souvik Bhattacherjee, Amit Chavan, Amol Deshpande, Aaron J. Elmore, Samuel Madden, and Aditya G. Parameswaran. Datahub: Collaborative data science & dataset version management at scale. In *CIDR 2015, Seventh Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 4-7, 2015, Online Proceedings*. www.cidrdb.org, 2015.

[5] Souvik Bhattacherjee, Amit Chavan, Silu Huang, Amol Deshpande, and Aditya Parameswaran. Principles of dataset versioning: Exploring the recreation/storage tradeoff. *Proc. VLDB Endow.*, 8(12):1346–1357, August 2015.

[6] Cristina De Castro, Fabio Grandi, and Maria Rita Scalas. Schema versioning for multitemporal relational databases. *Information Systems*, 22(5):249 – 290, 1997.

[7] Badrish Chandramouli, Johannes Gehrke, Jonathan Goldstein, Donald Kossmann, Justin J. Levandoski, Renato Marroquin, and Wenlei Xie. READY: completeness is in the eye of the beholder. In *CIDR 2017, 8th Biennial Conference on Innovative Data Systems Research, Chaminade, CA, USA, January 8-11, 2017, Online Proceedings*. www.cidrdb.org, 2017.

[8] Paul Clements and Linda Northrop. *Software Product Lines: Practices and Patterns*. Addison-Wesley, Boston, MA, 2001.

[9] AnHai Doan, Alon Halevy, and Zachary Ives. *Principles of Data Integration*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2012.

[10] Mina Farid, Alexandra Roatis, Ihab F. Ilyas, Hella-Franziska Hoffmann, and Xu Chu. Clams: Bringing quality to data lakes. In *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16, page 2089–2092, New York, NY, USA, 2016. Association for Computing Machinery.

[11] Kai Herrmann, Jan Reimann, Hannes Voigt, Birgit Demuth, Stefan Fromm, Robert Stelzmann, and Wolfgang Lehner. Database evolution for software product lines. In *DATA*, 2015.

[12] Silu Huang, Liqi Xu, Jialin Liu, Aaron J. Elmore, and Aditya Parameswaran. Orpheusdb: Bolt-on versioning for relational databases. *Proc. VLDB Endow.*, 10(10):1130–1141, June 2017.

[13] Edwin McKenzie and Richard Thomas Snodgrass. Schema evolution and the relational algebra. *Inf. Syst.*, 15(2):207–232, May 1990.

[14] Hyun J. Moon, Carlo A. Curino, Alin Deutsch, Chien-Yi Hou, and Carlo Zaniolo. Managing and querying transaction-time databases under schema evolution. *Proc. VLDB Endow.*, 1(1):882–895, August 2008.

[15] Martin Schäler, Thomas Leich, Marko Rosenmüller, and Gunter Saake. Building information system variants with tailored database schemas using features. In Jolita Ralyté, Xavier Franch, Sjaak Brinkkemper, and Stanislaw Wrycza, editors, *Advanced Information Systems Engineering*, pages 597–612, Berlin, Heidelberg, 2012. Springer.

[16] Norbert Siegmund, Christian Kästner, Marko Rosenmüller, Florian Heidenreich, Sven Apel, and Gunter Saake. Bridging the Gap Between Variability in Client Application and Database Schema. In *13. GI-Fachtagung Datenbanksysteme für Business, Technologie und Web (BTW)*, pages 297–306. Gesellschaft für Informatik (GI), 2009.

[17] Richard Thomas Snodgrass. *The TSQL2 Temporal Query Language*. Kluwer Academic Publishers, USA, 1995.

[18] Micheal Stonebraker, Dong Deng, and Micheal L. Brodie. Database decay and how to avoid it. In *Big Data (Big Data), 2016 IEEE International Conference*. IEEE, 2016.

[19] U. Störl, D. Müller, A. Tekleab, S. Tolale, J. Stenzel, M. Klettke, and S. Scherzinger. Curating variational data in application development. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 1605–1608, 2018.

APPENDICES

Appendix A: Variational Database Usecases

A.1 Variation in Space

[enron email usecase]

A.2 Variation in Time

[employee evolution usecase]