Generating Plans from Proofs

The Interpolation-based Approach to Query Reformulation

Synthesis Lectures on Data Management

Editor
Z. Meral Özsoyoğlu, ***Case Western Reserve University***

Synthesis Lectures on Data Management is edited by Meral Özsoyoğlu of Case Western Reserve University. The series publishes 80- to 150-page publications on topics pertaining to data management. Topics include query languages, database system architectures, transaction management, data warehousing, XML and databases, data stream systems, wide scale data distribution, multimedia data management, data mining, and related subjects.

Generating Plans from Proofs: The Interpolation-based Approach to Query Reformulation
Michael Benedikt, Julien Leblay, Balder ten Cate, and Efthymia Tsamoura
2015

Veracity of Data: From Truth Discovery Computation Algorithms to Models of Misinformation Dynamics
Laure Berti-Équille and Javier Borge-Holthoefer
2015

Datalog and Logic Databases
Sergio Greco and Cristina Molinaro
2015

Big Data Integration
Xin Luna Dong and Divesh Srivastava
2015

Instant Recovery with Write-Ahead Logging: Page Repair, System Restart, and Media Restore
Goetz Graefe, Wey Guy, and Caetano Sauer
2014

Similarity Joins in Relational Database Systems
Nikolaus Augsten and Michael H. Böhlen
2013

Information and Influence Propagation in Social Networks
Wei Chen, Laks V.S. Lakshmanan, and Carlos Castillo
2013

Data Cleaning: A Practical Perspective
Venkatesh Ganti and Anish Das Sarma
2013

Data Processing on FPGAs
Jens Teubner and Louis Woods
2013

Perspectives on Business Intelligence
Raymond T. Ng, Patricia C. Arocena, Denilson Barbosa, Giuseppe Carenini, Luiz Gomes, Jr., Stephan Jou, Rock Anthony Leung, Evangelos Milios, Renée J. Miller, John Mylopoulos, Rachel A. Pottinger, Frank Tompa, and Eric Yu
2013

Semantics Empowered Web 3.0: Managing Enterprise, Social, Sensor, and Cloud-based Data and Services for Advanced Applications
Amit Sheth and Krishnaprasad Thirunarayan
2012

Data Management in the Cloud: Challenges and Opportunities
Divyakant Agrawal, Sudipto Das, and Amr El Abbadi
2012

Query Processing over Uncertain Databases
Lei Chen and Xiang Lian
2012

Foundations of Data Quality Management
Wenfei Fan and Floris Geerts
2012

Incomplete Data and Data Dependencies in Relational Databases
Sergio Greco, Cristian Molinaro, and Francesca Spezzano
2012

Business Processes: A Database Perspective
Daniel Deutch and Tova Milo
2012

Data Protection from Insider Threats
Elisa Bertino
2012

Deep Web Query Interface Understanding and Integration
Eduard C. Dragut, Weiyi Meng, and Clement T. Yu
2012

P2P Techniques for Decentralized Applications
Esther Pacitti, Reza Akbarinia, and Manal El-Dick
2012

Query Answer Authentication
HweeHwa Pang and Kian-Lee Tan
2012

Declarative Networking
Boon Thau Loo and Wenchao Zhou
2012

Full-Text (Substring) Indexes in External Memory
Marina Barsky, Ulrike Stege, and Alex Thomo
2011

Spatial Data Management
Nikos Mamoulis
2011

Database Repairing and Consistent Query Answering
Leopoldo Bertossi
2011

Managing Event Information: Modeling, Retrieval, and Applications
Amarnath Gupta and Ramesh Jain
2011

Fundamentals of Physical Design and Query Compilation
David Toman and Grant Weddell
2011

Methods for Mining and Summarizing Text Conversations
Giuseppe Carenini, Gabriel Murray, and Raymond Ng
2011

Probabilistic Databases
Dan Suciu, Dan Olteanu, Christopher Ré, and Christoph Koch
2011

Peer-to-Peer Data Management
Karl Aberer
2011

Probabilistic Ranking Techniques in Relational Databases
Ihab F. Ilyas and Mohamed A. Soliman
2011

Uncertain Schema Matching
Avigdor Gal
2011

Fundamentals of Object Databases: Object-Oriented and Object-Relational Design
Suzanne W. Dietrich and Susan D. Urban
2010

Advanced Metasearch Engine Technology
Weiyi Meng and Clement T. Yu
2010

Web Page Recommendation Models: Theory and Algorithms
Sule Gündüz-Ögüdücü
2010

Multidimensional Databases and Data Warehousing
Christian S. Jensen, Torben Bach Pedersen, and Christian Thomsen
2010

Database Replication
Bettina Kemme, Ricardo Jimenez-Peris, and Marta Patino-Martinez
2010

Relational and XML Data Exchange
Marcelo Arenas, Pablo Barcelo, Leonid Libkin, and Filip Murlak
2010

User-Centered Data Management
Tiziana Catarci, Alan Dix, Stephen Kimani, and Giuseppe Santucci
2010

Data Stream Management
Lukasz Golab and M. Tamer Özsu
2010

Access Control in Data Management Systems
Elena Ferrari
2010

An Introduction to Duplicate Detection
Felix Naumann and Melanie Herschel
2010

Generating Plans from Proofs: The Interpolation-based Approach to Query Reformulation
Michael Benedikt, Julien Leblay, Balder ten Cate, and Efthymia Tsamoura

ISBN: 978-3-031-00728-6 paperback
ISBN: 978-3-031-01856-5 ebook

DOI 10.1007/978-3-031-01856-5

A Publication in the Spring series
SYNTHESIS LECTURES ON DATA MANAGEMENT

Lecture #43
Series Editor: Z. Meral Özsoyoğlu, *Case Western Reserve University*
Founding Editor: M. Tamer Özsu, *University of Waterloo*
Series ISSN
Print 2153-5418 Electronic 2153-5426

Generating Plans from Proofs

The Interpolation-based Approach to Query Reformulation

Michael Benedikt
Oxford University

Julien Leblay
National Institute of Advanced Industrial Science and Technology (AIST), Tokyo

Balder ten Cate
Google, Inc.

Efthymia Tsamoura
Oxford University

SYNTHESIS LECTURES ON DATA MANAGEMENT #43

ABSTRACT

Query reformulation refers to a process of translating a source query—a request for information in some high-level logic-based language—into a target plan that abides by certain interface restrictions. Many practical problems in data management can be seen as instances of the reformulation problem. For example, the problem of translating an SQL query written over a set of base tables into another query written over a set of views; the problem of implementing a query via translating to a program calling a set of database APIs; the problem of implementing a query using a collection of web services.

In this book we approach query reformulation in a very general setting that encompasses all the problems above, by relating it to a line of research within mathematical logic. For many decades logicians have looked at the problem of converting "implicit definitions" into "explicit definitions," using an approach known as interpolation. We will review the theory of interpolation, and explain its close connection with query reformulation. We will give a detailed look at how the interpolation-based approach is used to generate translations between logic-based queries over different vocabularies, and also how it can be used to go from logic-based queries to programs.

KEYWORDS

data integration, query optimization, query reformulation, views, tableau, Craig interpolation, Beth definability

Contents

Preface

Query reformulation. Query reformulation refers to a process of translating a *declarative source query* into a *target plan* that abides by certain *interface restrictions*, restrictions that the source query may not satisfy. By a source query we mean some request for information in a high-level logic-based language. For example a query asking for the names of the advisors of a university student called "Smith" would be written in the standard database language SQL as

$$\begin{array}{l}\mathsf{SELECT\ profname\ FROM\ Professor,\ Student}\\ \mathsf{WHERE\ Student.advisorid} = \mathsf{Professor.profid}\\ \mathsf{AND\ Student.lname} = \text{“}\mathsf{Smith}\text{”}\end{array}$$

and in first-order logic as:

$$\{\mathsf{profname} \mid \exists \mathsf{profid}\ \exists \mathsf{dname}\ \exists \mathsf{studid}\ \mathsf{Professor}(\mathsf{profid}, \mathsf{profname}, \mathsf{dname}) \wedge \mathsf{Student}(\mathsf{studid}, \text{“}\mathsf{Smith}\text{”}, \mathsf{profid})\}$$

Here it is assumed that the user posing the query thinks of the information in terms of two tables, Student and Professor. Student contains the student id and last name of each student along with the id of their advisor, while Professor contains entries for the id, last name, and department of each professor.

What kind of translation might we perform on an expression like the one above? It might be that to answer the source query it is necessary to access information stored in a different format. The stored data may have a table $\mathsf{Professor}'$ where the professor's id attribute is dropped, and a table $\mathsf{Student}'$ where the advisor's id is replaced with an attribute advisorname giving the advisor's last name. In order to retrieve the information over these reformatted sources, the query should be transformed. It is easy to see that in this case the correct transformation is just to get the advisorname attribute of rows corresponding to "Smith" in $\mathsf{Student}'$. In SQL the translation would be:

$$\mathsf{SELECT\ advisorname\ FROM\ Student'\ WHERE\ Student'.lname} = \text{“}\mathsf{Smith}\text{”}$$

and in first-order logic it would be:

$$\{\mathsf{advisorname} \mid \exists \mathsf{studid}\ \mathsf{Student}'(\mathsf{studid}, \text{“}\mathsf{Smith}\text{”}, \mathsf{advisorname})\}$$

In order to say that this represents a correct translation of the source query we need to know something about the semantics of the data. For us this will be captured by *integrity constraints*. In the above example, integrity constraints would describe the relationship between the accessible tables (Student′ and Professor′) and the tables mentioned in the source query (Student and Professor). Relative to those constraints, the SQL and logic translations above are correct.

Our notion of a target plan is very broad. We could be translating from one high-level query to another, as in the example above. We also consider translations from a high-level query to something operational, like a low-level program that makes calls to data access APIs. A basic function of a database management system is to translate a high-level language (e.g. first-order logic) to a low-level program. The goal there is to produce not just any equivalent program, but an efficient one. We will therefore look at the impact of efficiency considerations on reformulation. How to measure the efficiency of plans will not be our concern here—there is a rich research literature on the subject. We will instead be interested in algorithms that can return low-cost plans without specialized knowledge of the cost functions.

Reformulation via interpolation. Reformulating queries over restricted interfaces may sound very remote from concerns in mathematics. But it turns out that this problem is closely connected to a long line of research within mathematical logic. This book will provide an overview of the connection, explaining how ideas from logic can solve all of the reformulation problems above (and more). For each type of reformulation we will isolate a *semantic property* that any input query Q must have with respect to the target language and integrity constraints in order for the desired reformulation to exist. We then express this property as a *proof goal*: a statement that one logical formula follows from another. We will explain how to translate reformulation tasks into proof goals.

Reformulation proceeds by searching for a proof that witnesses the goal. From the proof we will then extract an *interpolant*, a logical formula that contains "only the necessary information" for the proof. We show that interpolants can be converted into reformulations through a very simple algorithm.

This "recipe" for reformulation dates back to work of the logician William Craig in the late 1950s. We show that it applies to a wide variety of reformulation scenarios. It is not a magic bullet that can always produce practical reformulation algorithms, but it often provides algorithms with optimal worst-case complexity, and it can be coupled with techniques for proof search and minimization of reformulations to become competitive with other reformulation techniques. We will explain the interpolation-based approach first for vocabulary-based restrictions, then for access-method based restrictions, and finally in the presence of cost information. We proceed in each case by explaining how the method is applied, then proving theorems stating that the resulting technique is complete—if a reformulation exists, the method will find it—and finally analyzing the worst-case complexity of the resulting algorithms.

About the book. This book has a number of objectives. It aims to explain formally what the interpolation-based method is, to exhibit the diverse ways in which it can be applied, and to explain the properties of the reformulations produced by the method. We also want to relate the interpolation-based approach to prior work on generating implementations from high-level queries.

This book has the most obvious interest to theoretically minded computer scientists. The focus throughout is on theorems: characterizations of reformulation (e.g., when does a source query have a reformulation of a certain kind?), expressiveness results (can a source query have a reformulation in one class, but no reformulation in another class?), and complexity bounds (what is the complexity of finding a reformulation in a certain class?). We connect our theorems to lines of research within a number of communities within theoretical computer science, particularly database theory, finite model theory, and knowledge representation. In a few cases, we state a theorem but omit the verification, pointing the reader to a paper where the full proofs appear. But the main results are proven in detail in order to present the theory in a self-contained manner. For many of the results, complete proofs have never appeared in print prior to this work.

A second audience for the book consists of researchers in logic. They will be very familiar with basic results about interpolation, along with the related topic of going from implicit definitions to explicit ones, but perhaps not with either the theory or the practice of databases. We aim to introduce logicians to the application of interpolation in data management. We hope that the results here give a new constructive perspective on the relationship between syntax and semantics, a major theme of research in both theoretical computer science and model theory. This book can be seen as working out more practical consequences of what are called "preservation theorems" in first-order model theory —theorems that characterize subclasses of first-order logic via semantic properties.

Finally, we hope that parts of the text will be of interest to researchers in databases, even those who do not work in theory. Chapter 4 and Chapter 5 are the most accessible parts of the text for researchers in data integration and query optimization with a more applied background. These two chapters deal with algorithms that can be understood without reference to interpolation, and without a background in first-order logic.

In trying to give a comprehensive picture of the theory of reformulation, we have completely omitted a host of issues that are critical in practice. For example:

- We deal only with set semantics for queries, not the bag semantics used in SQL.
- We consider only first-order queries, without considering aggregates like COUNT and SUM that play a crucial role in many database applications.
- Our model of data is "un-typed," assuming every column takes values from a fixed infinite set. We assume this infinite set has no structure that can be referenced in queries or constraints. Thus we do not allow queries and constraints that can mention integer inequality or arithmetic, string concatenation or substring comparisons, all of which appear in constraints and queries in practice.
- We do not cover all the integrity constraints that are important in practice. We present some general results about reformulation with arbitrary first-order logic constraints, which

are applicable to all common SQL schema constraints, including referential constraints and key constraints. We obtain decidability and complexity results for reformulation, for some limited constraint classes. But we omit an analysis of a few classes that are significant for database applications. For example, we do not give any special attention to equality-generating dependencies, which subsume the key constraints that play a fundamental role in SQL.

- We consider the problem of getting low-cost reformulations, but our theoretical results apply only to the case of very simplistic cost functions. We do not analyze realistic cost functions that are used in the database or the web data integration setting.

Many of these pragmatic issues are discussed in an earlier textbook [Toman and Weddell, 2011]. Others (like aggregation) represent difficult open problems for any theory of reformulation.

Although the book is focused on theory, we try to give a sense of how the interpolation-based framework is useful in practice. Thus throughout the book we present examples of the results in (simplified) application scenarios, and give pointers to further work concerning systems based on the theory.

Michael Benedikt, Julien Leblay, Balder ten Cate, and Efthymia Tsamoura
February 2016

Acknowledgments

The authors are very grateful to the following people for reviewing early copies of the manuscript and providing extensive comments: Antoine Amarilli, Cristian Riveros, Michael Vanden Boom, and James Worrell. We are also thankful to Carsten Lutz, Jan Van den Bussche, and Victor Vianu for their detailed feedback on the first official draft of the book. Benedikt's work was sponsored by the Engineering and Physical Sciences Research Council of the United Kingdom, grants EP/M005852/1 and EP/L012138/1.

Michael Benedikt, Julien Leblay, Balder ten Cate, and Efthymia Tsamoura
February 2016