Foundations of DATA EXCHANGE

Marcelo Arenas, Pablo Barceló,
Leonid Libkin, and Filip Murlak

CAMBRIDGE

FOUNDATIONS OF DATA EXCHANGE

MARCELO ARENAS
Pontificia Universidad Católica de Chile

PABLO BARCELÓ
Universidad de Chile

LEONID LIBKIN
University of Edinburgh

FILIP MURLAK
Uniwersytet Warszawski, Poland

CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE
UNIVERSITY PRESS

University Printing House, Cambridge CB2 8BS, United Kingdom

Published in the United States of America by Cambridge University Press, New York

Cambridge University Press is part of the University of Cambridge.

It furthers the University's mission by disseminating knowledge in the pursuit of education, learning and research at the highest international levels of excellence.

www.cambridge.org
Information on this title: www.cambridge.org/9781107016163

First published 2014

Printed in the United Kingdom by CPI Group Ltd, Croydon CR0 4YY

A catalogue record for this publication is available from the British Library

Library of Congress Cataloguing in Publication data

ISBN 978-1-107-01616-3 Hardback

FOUNDATIONS OF DATA EXCHANGE

The problem of exchanging data between different databases with different schemas is an area of immense importance. Consequently data exchange has been one of the most active research topics in databases over the past decade. Foundational questions related to data exchange largely revolve around three key problems: how to build target solutions; how to answer queries over target solutions; and how to manipulate schema mappings themselves? The last question is also known under the name "metadata management", since mappings represent metadata, rather than data in the database.

In this book the authors summarize the key developments of a decade of research. Part I introduces the problem of data exchange via examples, both relational and XML; Part II deals with exchanging relational data; Part III focuses on exchanging XML data; and Part IV covers metadata management.

MARCELO ARENAS is an Associate Professor at the Department of Computer Science at the Pontificia Universidad Catolica de Chile. He received his Ph.D. from the University of Toronto in 2005. His research interests are in different aspects of database theory, such as expressive power of query languages, database semantics, inconsistency handling, database design, XML databases, data exchange, metadata management and database aspects of the Semantic Web. He has received an IBM Ph.D. Fellowship (2004), seven best paper awards (PODS 2003, PODS 2005, ISWC 2006, ICDT 2010, ESWC 2011, PODS 2011 and WWW 2012) and an ACM-SIGMOD Dissertation Award Honorable Mention in 2006 for his Ph.D. dissertation "Design Principles for XML Data". He has served on multiple program committees, and since 2009 he has been participating as an invited expert in the World Wide Web Consortium.

PABLO BARCELÓ is an Assistant Professor in the Department of Computer Science at the University of Chile. He received his Ph.D. from the University of Toronto in 2006. His main research interest is in the area of Foundations of Data Management, in particular, query languages, data exchange, incomplete databases, and, recently, graph databases. He has served on program committees of some of the major conferences in database theory and the theoretical aspects of Computer Science (PODS, ICDT, CIKM, STACS, SIGMOD).

LEONID LIBKIN is Professor of Foundations of Data Management in the School of Informatics at the University of Edinburgh. He was previously a Professor at the University of Toronto and a member of research staff at Bell Laboratories in Murray Hill. He received his PhD from the University of Pennsylvania in 1994. His main research interests are in the areas of data management and applications of logic in computer science. He has written four books and over 150 technical papers. He was the recipient of a Marie Curie Chair Award from the EU in 2006, and won four best paper awards. He has chaired programme committees of major database conferences (ACM PODS, ICDT) and was the conference chair of the 2010 Federated Logic Conference. He has given many invited conference talks and has served on multiple program committees and editorial boards. He is an ACM fellow and a fellow of the Royal Society of Edinburgh.

FILIP MURLAK is assistant professor at the Faculty of Mathematics, Informatics, and Mechanics at the University of Warsaw, Poland. Previously he was research fellow at the University of Edinburgh. He received his PhD from the University of Warsaw in 2008. His main research areas are automata theory and semi-structured data. He was the recipient of the best paper award at ICALP 2006, the Witold Lipski Prize for young researchers in 2008, and the Homing Plus scholarship from the Foundation for Polish Science in 2010. He was co-chair of MFCS 2011 and served on program committees of several database and theoretical computer science conferences.

To Magdalena, Marcelito and Vanny, for their love and support.
M. A.

To Salvador, Gabi and my parents.
P. B.

To my parents.
L. L.

To my grandparents, who chose to send their kids to university.
F. M.

Preface

Data exchange, as the name suggests, is the problem of exchanging data between different databases that have different schemas. One often needs to exchange data between existing legacy databases, whose schemas cannot be easily modified, and thus one needs to specify rules for translating data from one database to the other. These rules are known as schema mappings. Once a source database and a schema mapping are given, one needs to transfer data to the target, i.e., construct a target database. And once the target database is constructed, one needs to answer queries against it.

This problem is quite old; it has been studied, and systems have been built, but it was done in a rather ad hoc way. A systematic study of the problem of data exchange commenced with the 2003 paper "Data exchange: semantics and query answering" by Fagin, Kolaitis, Miller, and Popa, published in the proceedings of the International Conference on Database Theory. A large number of followup papers appeared, and for a while data exchange was one of the most active research topics in databases. Foundational questions related to data exchange largely revolved around three key problems:

1. how to build a target solution;
2. how to answer queries over target solutions; and
3. how to manipulate schema mappings themselves.

The last question is also known under the name of metadata management, since mappings represent metadata, rather than data in the database.

This book summarizes the key developments of the decade of research in the area of data exchange. It is organized into four parts.

In Part One, the problem of data exchange is introduced via examples, both relational and XML. We present key definitions: of schema mappings, of solutions (possible target instances), and of query answering and rewriting. We also describe some background material on relational databases, query languages, incomplete data, complexity theory, and automata theory (that will be required in the study of XML data exchange).

Part Two deals with exchanging relational data. We start by looking at the problem of checking if solutions, or possible target instances, exist. In general, the problem may even be undecidable, so we look at restrictions on schema mappings to guarantee not only

decidability but also tractability of the problem of building solutions. Under one restriction, called weak acyclicity, particularly nice solutions can be constructed efficiently. These are called universal solutions, and they are particularly well suited for query answering in data exchange. It turns out that a given source may have many targets compatible with it, and thus we use the semantics of certain answers, i.e., answers true in all compatible targets. It is those answers that can be efficiently found in universal solutions. Universal solutions themselves are not unique, and we look at two types of these, the canonical universal solution, and the core, that have particularly nice properties. We also look at alternative ways of defining the semantics of query answering in data exchange.

Part Three deals with exchanging XML data. It mimics the developments of Part Two, but there are crucial differences between XML and relations. In particular, the complexity of many basic tasks increases in the XML case, and one needs different types of restrictions for keeping the complexity manageable. We identify those, particularly by placing restrictions on schemas, as it is their complexity that affects data exchange problems most. We also look at answering two types of queries: XML-to-relations, and XML-to-XML. Finally, we show how to perform XML data exchange tasks using relational data exchange engines.

Part Four deals with metadata management, i.e., handling schema mappings themselves. We deal with their static analysis, in particular, the consistency (or satisfiability) problem and the simplification problem. We study schema evolution described by means of operations on mappings. Two key operations one needs are composition of mappings and inverting mappings, and we provide a detailed study of both.

Each part of the book comes with a summary, bibliographic comments, and exercises.

A much shorter draft of this book was published in the Morgan & Claypool Synthesis series under the title *"Relational and XML Data Exchange"* in 2010. While working on that short version, and on the full draft, as well as on papers that are reflected in this book, we benefited from comments and critical remarks from our colleagues. We would like to thank Shunichi Amano, Mikołaj Bojańczyk, Rada Chirkova, Wojtek Czerwiński, Claire David, Ronald Fagin, Wenfei Fan, Amélie Gheerbrant, Andre Hernich, Phokion Kolaitis, Maurizio Lenzerini, Katja Losemann, Wim Martens, Jorge Pérez, Juan Reutter, Cristian Riveros, Miguel Romero, Nicole Schweikardt, Thomas Schwentick, and Cristina Sirangelo. We are also grateful to Tamer Özsu and Diane Cerra for convincing us to write the short Morgan & Claypool version of the book, and to David Tranah at Cambridge University Press for persuading us to turn it into a proper book, and for his patience and assistance.

Santiago, Edinburgh, Warsaw M.A., P.B., L.L., F.M.

Contents

Preface *page* xi

PART ONE GETTING STARTED 1

1 Data exchange by example 3
1.1 A data exchange example 3
1.2 Overview of the main tasks in data exchange 9
1.3 Data exchange vs data integration 11

2 Theoretical background 12
2.1 Relational database model 12
2.2 Query languages 14
2.3 Incomplete data 18
2.4 Complexity classes 21
2.5 Basics of automata theory 27

3 Data exchange: key definitions 29
3.1 Schema mappings 29
3.2 Solutions 30
3.3 Query answering and rewriting 31
3.4 Bibliographic comments 32

PART TWO RELATIONAL DATA EXCHANGE 33

4 The problem of relational data exchange 35
4.1 Key definitions 35
4.2 Key problems 39

5 Existence of solutions 43
5.1 The problem and easy cases 43
5.2 Undecidability for st-tgds and target constraints 44
5.3 The chase 46

5.4 Weak acyclicity of target constraints 49
5.5 Complexity of the problem 53

6 Good solutions 56
6.1 Universal solutions 56
6.2 Existence of universal solutions 59
6.3 Canonical universal solution and chase 65
6.4 The core 68

7 Query answering and rewriting 75
7.1 Answering relational calculus queries 75
7.2 Answering conjunctive queries 76
7.3 Conjunctive queries with inequalities 78
7.4 Tractable query answering with negation 81
7.5 Rewritability over special solutions 88
7.6 Non-rewritability tool: locality 91

8 Alternative semantics 97
8.1 Universal solutions semantics 98
8.2 Closed-world semantics 102
8.3 Closed-world semantics and target constraints 112
8.4 Clopen-world semantics 121

9 Endnotes to Part Two 124
9.1 Summary 124
9.2 Bibliographic comments 125
9.3 Exercises 126

PART THREE XML DATA EXCHANGE 133

10 The problem of XML data exchange 135
10.1 XML documents and schemas 135
10.2 Key problems of XML data exchange 141

11 Patterns and mappings 143
11.1 Tree patterns: classification and complexity 143
11.2 XML schema mappings and their complexity 153

12 Building solutions 158
12.1 Building solutions revisited 158
12.2 A simple exhaustive search algorithm 159
12.3 Nested-relational DTDs 162
12.4 The algorithm for regular schemas 168
12.5 The general algorithm 172
12.6 Combined complexity of solution building 178

13 Answering tuple queries 182
13.1 The query answering problem 182
13.2 An upper bound 184
13.3 Sources of intractability 185
13.4 Tractable query answering 189

14 XML-to-XML queries 193
14.1 XML-to-XML query language TQL 193
14.2 Notion of certain answers 196
14.3 Certain answers for TQL queries 201
14.4 XML-to-XML queries in data exchange 203

15 XML data exchange via relations 206
15.1 Translations and correctness 206
15.2 Translations of schemas and documents 209
15.3 Translations of patterns, mappings and queries 213
15.4 Answering XML queries using relational data exchange 217

16 Endnotes to Part Three 220
16.1 Summary 220
16.2 Bibliographic comments 221
16.3 Exercises 221

PART FOUR METADATA MANAGEMENT 225

17 What is metadata management? 227
17.1 Reasoning about schema mappings 227
17.2 Manipulating schema mappings 228

18 Consistency of schema mappings 230
18.1 Problems statements 230
18.2 Consistency for XML 231
18.3 Absolute consistency for XML 238

19 Mapping composition 249
19.1 The notion of composition and key problems 249
19.2 Complexity of relational composition 251
19.3 Extending st-tgds with second-order quantification 253
19.4 Complexity of XML composition 261
19.5 Tractable XML composition 265

20 Inverting schema mappings 272
20.1 A first definition of inverse 272
20.2 Bringing exchanged data back: the recovery of a schema mapping 280
20.3 Computing the inverse operator 287
20.4 Inverses under extended semantics 294

21 Structural characterizations of schema mapping 300
21.1 Structural properties 300
21.2 Schema mapping languages characterizations 301
21.3 An application: simplifying schema mappings 307
22 Endnotes to Part Four 312
22.1 Summary 312
22.2 Bibliographic comments 313
22.3 Exercises 315

References 321
Index 327

PART ONE

GETTING STARTED

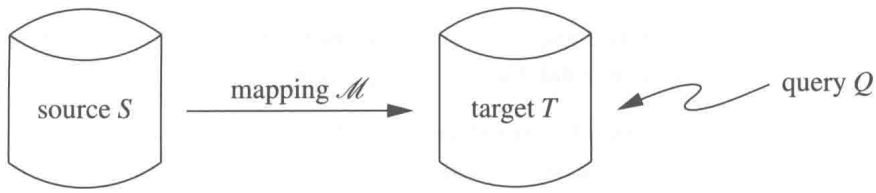
1

Data exchange by example

Data exchange is the problem of finding an instance of a target schema, given an instance of a source schema and a specification of the relationship between the source and the target. Such a target instance should correctly represent information from the source instance under the constraints imposed by the target schema, and should allow one to evaluate queries on the target instance in a way that is semantically consistent with the source data.

Data exchange is an old problem that re-emerged as an active research topic recently due to the increased need for exchange of data in various formats, often in e-business applications.

The general setting of data exchange is this:



We have fixed source and target schemas, an instance S of the source schema, and a mapping $\mathcal{M}$ that specifies the relationship between the source and the target schemas. The goal is to construct an instance T of the target schema, based on the source and the mapping, and answer queries against the target data in a way consistent with the source data.

The goal of this introductory chapter is to make precise some of the key notions of data exchange: schema mappings, solutions, source-to-target dependencies, and certain answers. We do it by means of an example we present in the next section.

1.1 A data exchange example

Suppose we want to create a database containing three relations:

- `ROUTES(flight#,source,destination)`

  This relation has information about routes served by several airlines: it has a `flight#` attribute (e.g., AF406 or KLM1276), as well as `source` and `destination` attributes (e.g., Paris and Santiago for AF406).

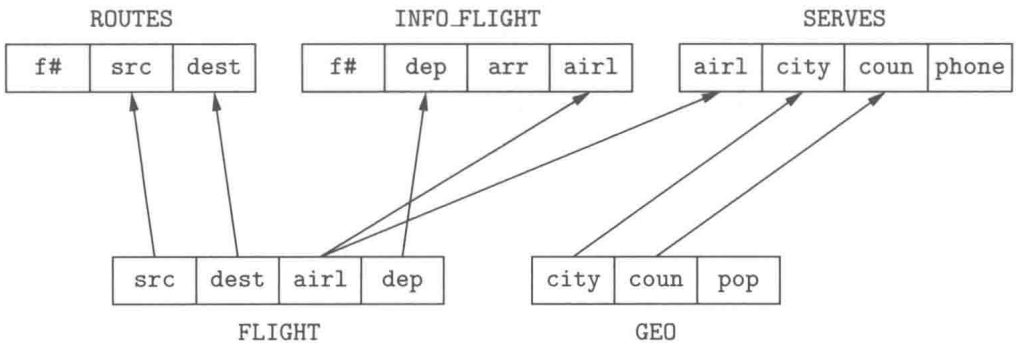


Figure 1.1 Schema mapping: a simple graphical representation

- `INFO_FLIGHT(flight#,departure_time,arrival_time,airline)`
  This relation provides additional information about the flight: departure and arrival times, as well as the name of an airline.
- `SERVES(airline,city,country,phone)`
  This relation has information about cities served by airlines: for example, it may have a tuple (*AirFrance*, *Santiago*, *Chile*, *5550000*), indicating that Air France serves Santiago, Chile, and its office there can be reached at 555-0000.

We do not start from scratch: there is a source database available from which we can transfer information. This source database has two relations:

- `FLIGHT(source,destination,airline,departure)`
  This relation contains information about flights, although not all the information needed in the target. We only have source, destination, and airline (but no flight number), and departure time (but no arrival time).
- `GEO(city,country,population)`
  This relation has some basic geographical information: cities, countries where they are located, and their population.

As the first step of moving the data from the source database into our target, we have to specify a *schema mapping*, a set of relationships between the two schemas. We can start with a simple graphical representation of such a mapping shown in Figure 1.1. The arrows in such a graphical representation show the relationship between attributes in different schemas.

But simple connections between attributes are not enough. For example, when we create records in `ROUTES` and `INFO_FLIGHT` based on a record in `FLIGHT`, we need to ensure that the values of the `flight#` attribute (abbreviated as `f#` in the figure) are the same. This is indicated by a curved line connecting these attributes. Likewise, when we populate table `SERVES`, we only want to include cities which appear in table `FLIGHT` – this is indicated by the line connecting attributes in tables `GEO` and `FLIGHT`.

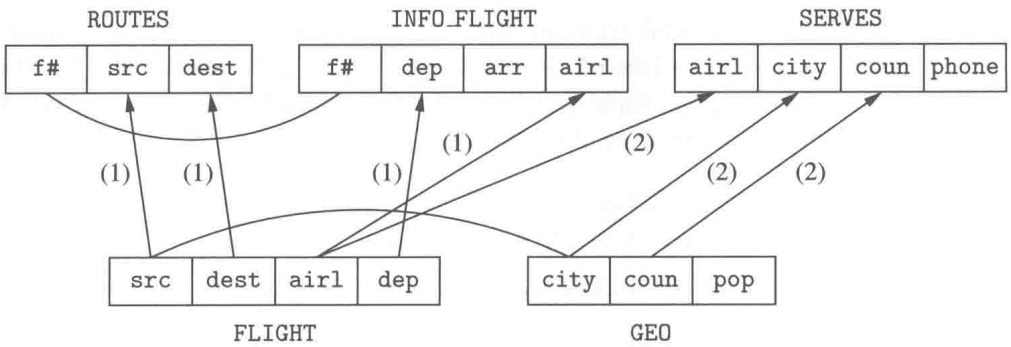


Figure 1.2 Schema mapping: a proper graphical representation

Furthermore, there are several *rules* in a mapping that help us populate the target database. In this example, we can distinguish two rules. One uses table FLIGHT to populate ROUTES and INFO_FLIGHT, and the other uses both FLIGHT and GEO to populate SERVES. So in addition we annotate arrows with names or numbers of rules that they are used in. Such a revised representation is shown in Figure 1.2.

While it might be easy for someone understanding source and target schemas to produce a graphical representation of the mapping, we need to translate it into a formal specification. Let us look at the first rule which says:

whenever we have a tuple (src,dest,airl,dep) *in relation* FLIGHT, *we must have a tuple in* ROUTES *that has* src *and* dest *as the values of the second and the third attributes, and a tuple in* INFO_FLIGHT *that has* dep *and* airl *as the second and the fourth attributes.*

Formally, this can be written as:

$$\texttt{FLIGHT(src,dest,airl,dep)} \longrightarrow \texttt{ROUTES(\_,src,dest), INFO\_FLIGHT(\_,dep,\_,airl).}$$

This is not fully satisfactory: indeed, as we lose information that the flight numbers must be the same; hence, we need to explicitly mention the names of all the variables, and produce the following rule:

$$\texttt{FLIGHT(src,dest,airl,dep)} \longrightarrow \texttt{ROUTES(f\#,src,dest), INFO\_FLIGHT(f\#,dep,arr,airl).}$$

What is the meaning of such a rule? In particular, what are those variables that appear in the target specification without being mentioned in the source part? What the mapping says is that values for these variables must *exist* in the target, in other words, the following must be satisfied:

$$\texttt{FLIGHT(src,dest,airl,dep)} \longrightarrow \exists \texttt{f\#}\, \exists \texttt{arr}\, \big(\texttt{ROUTES(f\#,src,dest)} \wedge \texttt{INFO\_FLIGHT(f\#,dep,arr,airl)} \big).$$