Computer-Assisted Query Formulation

Alvin Cheung
University of Washington
akcheung@cs.washington.edu

Armando Solar-Lezama
MIT CSAIL
asolar@csail.mit.edu

the essence of knowledge
Boston — Delft

Foundations and Trends® in Programming Languages

Published, sold and distributed by:
now Publishers Inc.
PO Box 1024
Hanover, MA 02339
United States
Tel. +1-781-985-4510
www.nowpublishers.com
sales@nowpublishers.com

Outside North America:
now Publishers Inc.
PO Box 179
2600 AD Delft
The Netherlands
Tel. +31-6-51115274

The preferred citation for this publication is

A. Cheung and A. Solar-Lezama. *Computer-Assisted Query Formulation.* Foundations and Trends® in Programming Languages, vol. 3, no. 1, pp. 1–94, 2016.

This Foundations and Trends® issue was typeset in LaTeX using a class file designed by Neal Parikh. Printed on acid-free paper.

ISBN: 978-1-68083-037-8

Full text available at: http://dx.doi.org/10.1561/2500000018

Foundations and Trends® in Programming Languages

Volume 3, Issue 1, 2016

Editorial Board

Editorial Scope

Topics

Foundations and Trends® in Programming Languages publishes survey and tutorial articles in the following topics:

- Abstract interpretation
- Compilation and interpretation techniques
- Domain specific languages
- Formal semantics, including lambda calculi, process calculi, and process algebra
- Language paradigms
- Mechanical proof checking
- Memory management
- Partial evaluation
- Program logic
- Programming language implementation
- Programming language security
- Programming languages for concurrency
- Programming languages for parallelism
- Program synthesis
- Program transformations and optimizations
- Program verification
- Runtime techniques for programming languages
- Software model checking
- Static and dynamic program analysis
- Type theory and type systems

Information for Librarians

Foundations and Trends® in Programming Languages, 2016, Volume 3, 4 issues. ISSN paper version 2325-1107. ISSN online version 2325-1131. Also available as a combined paper and online subscription.

Foundations and Trends® in Programming Languages
Vol. 3, No. 1 (2016) 1–94

DOI: 10.1561/2500000018

Computer-Assisted Query Formulation

Alvin Cheung
University of Washington
akcheung@cs.washington.edu

Armando Solar-Lezama
MIT CSAIL
asolar@csail.mit.edu

Contents

Abstract

Database management systems (DBMS) typically provide an application programming interface for users to issue queries using query languages such as SQL. Many such languages were originally designed for business data processing applications. While these applications are still relevant, two other classes of applications have become important users of data management systems: (a) web applications that issue queries programmatically to the DBMS, and (b) data analytics involving complex queries that allow data scientists to better understand their datasets. Unfortunately, existing query languages provided by database management systems are often far from ideal for these application domains.

In this tutorial, we describe a set of technologies that assist users in specifying database queries for different application domains. The goal of such systems is to bridge the gap between current query interfaces provided by database management systems and the needs of different usage scenarios that are not well served by existing query languages. We discuss the different interaction modes that such systems provide and the algorithms used to infer user queries. In particular, we focus on a new class of systems built using program synthesis techniques, and furthermore discuss opportunities in combining synthesis and other methods used in prior systems to infer user queries.

A. Cheung and A. Solar-Lezama. *Computer-Assisted Query Formulation*.
Foundations and Trends® in Programming Languages, vol. 3, no. 1, pp. 1–94, 2016.
DOI: 10.1561/2500000018.

1

Introduction

From financial transactions to online shopping, we interact with database management systems (DBMSs) on a daily basis. Since the initial development of relational database systems, various query languages such as SQL have been developed for users to interact with the DBMS. Many of these languages proved very effective for what was originally their primary application: business data processing (*e.g.*, generating transaction reports at a financial institution). Unfortunately, many important applications of DBMSs that have emerged in recent decades have proven to be a less than ideal fit for the interaction models supported by traditional DBMSs.

One particularly important extension to the business data processing application space corresponds to applications with complex business logic, such as social network websites, online shopping applications, *etc.* Unfortunately, traditional query interfaces often make developing such applications difficult. First, the general-purpose languages in which these applications are usually written (*e.g.*, Java or Python) are quite different from the query languages supported by the DBMS. This forces developers to learn a new language—and often a new programming paradigm altogether. For example, an application developer who is used to thinking about computation over objects stored in the program heap will need to recast her computation in terms of structured relations stored on disks when interacting with a DBMS. Moreover, in addition to being concerned about efficient memory layout for retrieving in-memory objects, she will also need to understand the costs associated with bringing objects into memory from the disk. This "impedance mismatch" [Copeland and Maier, 1984] problem has plagued application developers for decades. Today, this mismatch is often addressed by application frameworks known as Object Relational Mapping (ORM) Frameworks that eliminate the need to think in terms of two distinct programming models. Unfortunately, the use of ORMs often imposes significant performance costs [Subramanian].

There are many reasons for the performance cost of ORMs, but one that is especially significant is that they encourage a programming style where computation that could have been implemented with a single query and a single round trip to the database is instead implemented with several simpler queries connected together with imperative code that manipulates their results. This is problematic because in addition to increasing the number of round trips and the amount of data that needs to be transferred between the application and the DBMS, doing so also increases the cost of the computation, since the DBMS is in much better position to optimize queries compared to a general-purpose code compiler trying to optimize a block of imperative code that happens to implement a relational operation.

As an example, while a relational join between relations R and S can be implemented using a nested loop, with each loop processing tuples from the two respective relations fetched from the DBMS, it is much more efficient to implement the join as a single SQL query, as the DBMS can choose the best way to implement the join during query optimization.

In this tutorial we focus on a new approach based on verified lifting [Cheung et al., 2015] to reduce the performance cost of these application frameworks, allowing programmers to enjoy the benefits of the reduced impedance mismatch. The first step in this technique is to identify places in the application code where the programmer is using imperative code to implement functionality that could be implemented as part of a query. The second and most important step is to use program synthesis technology to derive a query that is provably equivalent to the imperative code. Once that is done, the third step involves generating a new version of the code that uses the query in place of the original code.

The technology behind this work was originally published earlier [Cheung et al., 2013]. In this paper, we expand on the content of that original paper in order to make the technology more accessible to researchers without a strong background in program synthesis or verification, as well as to researchers who may not be as familiar with database concepts. In Section 2, we provide a quick primer on query execution and query processing, focusing on key concepts that will help the reader understand the reasons for the performance problems introduced by ORMs. Section 3 provides a comprehensive primer on program synthesis technology, focusing in particular on the techniques that are leveraged by QBS, and putting them in context of other synthesis technologies. Section 4 describes the details of the QBS approach, and finally Section 5 describes the state of the art in terms of applications of synthesis to interact with DBMS systems and promising directions for future work.

References

Azza Abouzied, Joseph M. Hellerstein, and Avi Silberschatz. Playful query specification with DataPlay. *VLDB Endowment*, 5(12):1938–1941, 2012.

Azza Abouzied, Dana Angluin, Christos H. Papadimitriou, Joseph M. Hellerstein, and Avi Silberschatz. Learning and verifying quantified boolean queries by example. In *ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 49–60, 2013.

John R. Allen and Ken Kennedy. Automatic loop interchange. In *International Conference on Compiler Construction*, pages 233–246, 1984.

Rajeev Alur, Rastislav Bodík, Garvit Juniwal, Milo M. K. Martin, Mukund Raghothaman, Sanjit A. Seshia, Rishabh Singh, Armando Solar-Lezama, Emina Torlak, and Abhishek Udupa. Syntax-guided synthesis. In *Formal Methods in Computer-Aided Design*, pages 1–17, 2013.

Ion Androutsopoulos, Graeme D. Ritchie, and Peter Thanisch. Natural language interfaces to databases - an introduction. *Natural Language Engineering*, 1(1):29–81, 1995.

Dana Angluin, Michael Frazier, and Leonard Pitt. Learning conjunctions of horn clauses. *Machine Learning*, 9:147–164, 1992.

Mark Bickford, Christoph Kreitz, Robbert Van Renesse, and Robert Constable. An experiment in formal design using meta-properties. In *DISCEX-II: The 2nd DARPA Information Survivability Conference and Exposition. IEEE*, 2001.

Andreas Blass and Yuri Gurevich. Inadequacy of computable loop invariants. 2(1):1–11, 2001.

Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

Mark Burstein, Drew Mcdermott, Douglas R. Smith, and Stephen J. Westfold. Derivation of glue code for agent interoperation. In *4th Intl. Conf. on Autonomous Agents*, pages 277–284, 2000.

Ugur Çetintemel, Mitch Cherniack, Justin DeBrabant, Yanlei Diao, Kyriaki Dimitriadou, Alexander Kalinin, Olga Papaemmanouil, and Stanley B. Zdonik. Query steering for interactive data exploration. In *Biennial Conference on Innovative Data Systems Research*, 2013.

Gloria Chatzopoulou, Magdalini Eirinaki, and Neoklis Polyzotis. Query recommendations for interactive database exploration. In *International Conference on Scientific and Statistical Database Management*, pages 3–18, 2009.

Alvin Cheung, Arvind Thiagarajan, and Samuel Madden. Automatically generating interesting events with LifeJoin. In *International Conference on Embedded Networked Sensor Systems*, pages 411–412, 2011.

Alvin Cheung, Armando Solar-Lezama, and Samuel Madden. Using program synthesis for social recommendations. In *International Conference on Information and Knowledge Management*, pages 1732–1736, 2012.

Alvin Cheung, Armando Solar-Lezama, and Samuel Madden. Optimizing database-backed applications with query synthesis. In *ACM SIGPLAN Conf. on Programming Language Design and Implementation*, pages 3–14, 2013.

Alvin Cheung, Samuel Madden, and Armando Solar-Lezama. Sloth: being lazy is a virtue (when issuing database queries). In *ACM SIGMOD International Conference on Management of Data*, pages 931–942, 2014.

Alvin Cheung, Shoaib Kamil, and Armando Solar-Lezama. Bridging the gap between general-purpose and domain-specific compilers with synthesis. In *1st Summit on Advances in Programming Languages*, pages 51–62, 2015.

David A. Cieslak and Nitesh V. Chawla. Learning decision trees for unbalanced data. In *European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 241–256, 2008.

Edmund M. Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. Counterexample-guided abstraction refinement. In *Computer Aided Verification*, pages 154–169, 2000.

E. F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, 1970.

Ezra Cooper, Sam Lindley, Philip Wadler, and Jeremy Yallop. Links: Web programming without tiers. In *International Conference on Formal Methods for Components and Objects*, pages 266–296, 2007.

George Copeland and David Maier. Making smalltalk a database system. In *ACM SIGMOD International Conference on Management of Data*, pages 316–325, 1984.

Chris J. Date. *An introduction to database systems (7th ed.)*. Addison-Wesley-Longman, 2000.

David Déharbe, Pascal Fontaine, Stephan Merz, and Bruno Woltzenlogel Paleo. Exploiting symmetry in SMT problems. In *International Conference on Automated Deduction*, pages 222–236, 2011.

Edsger W. Dijkstra. Guarded commands, nondeterminacy and formal derivation of programs. *Communications of the ACM*, 18(8):453–457, 1975.

Kyriaki Dimitriadou, Olga Papaemmanouil, and Yanlei Diao. Explore-by-example: an automatic query steering framework for interactive data exploration. In *ACM SIGMOD International Conference on Management of Data*, pages 517–528, 2014.

Django. http://www.djangoproject.com. Accessed: 2014-11-22.

Robert Floyd. Assigning meanings to programs. *American Mathematical Society Symposia on Applied Mathematics*, 19:19–31, 1967.

Eibe Frank and Ian H. Witten. Generating accurate rule sets without global optimization. In *International Conference on Machine Learning*, pages 144–151, 1998.

David Gries. *The Science of Programming*. Springer, 1987.

Sumit Gulwani and Mark Marron. NLyze: interactive programming by natural language for spreadsheet data analysis and manipulation. In *ACM SIGMOD International Conference on Management of Data*, pages 803–814, 2014.

Sumit Gulwani, Susmit Jha, Ashish Tiwari, and Ramarathnam Venkatesan. Synthesis of loop-free programs. In *ACM SIGPLAN Conf. on Programming Language Design and Implementation*, pages 62–73, 2011.

Sumit Gulwani, William R. Harris, and Rishabh Singh. Spreadsheet data manipulation using examples. *Communications of the ACM*, 55(8):97–105, August 2012.

Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

C. A. R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580, 1969.

International Organization for Standardization. Information technology – Database languages – SQL – Part 3: Call-Level Interface (SQL/CLI). ISO ISO/IEC 9075-3:2008, Geneva, Switzerland, 2008.

International Organization for Standardization. Information technology – Database languages – SQL – Part 4: Persistent Stored Modules (SQL/PSM). ISO ISO/IEC 9075-4:2011, Geneva, Switzerland, 2011.

itracker Issue Management System. http://itracker.sourceforge.net/index.html. Accessed: Apr 5, 2016.

Ming-Yee Iu and Willy Zwaenepoel. HadoopToSQL: a mapReduce query optimizer. In *European Conference on Computer systems*, pages 251–264, 2010.

Ming-Yee Iu, Emmanuel Cecchet, and Willy Zwaenepoel. JReq: Database queries in imperative languages. In *International Conference on Compiler Construction*, pages 84–103, 2010.

Java Persistence 2.0 Expert Group. Java Persistence API. http://jcp.org/aboutJava/communityprocess/final/jsr317, 2009.

JBoss. Hibernate documentation. http://www.hibernate.org. Accessed: 2014-11-22.

JDBC 4.2 Expert Group. JDBC 4.2 API. http://jcp.org/aboutJava/communityprocess/mrel/jsr221, 2014.

Susmit Jha, Sumit Gulwani, Sanjit A. Seshia, and Ashish Tiwari. Oracle-guided component-based program synthesis. In *International Conference on Software Engineering*, pages 215–224, 2010.

Lilong Jiang, Michael Mandel, and Arnab Nandi. GestureQuery: A multi-touch database query interface. *VLDB Endowment*, 6(12):1342–1345, August 2013.

jOOQ. http://www.jooq.org. Accessed: 2014-11-22.

Shoaib Kamil, Alvin Cheung, Shachar Itzhaky, and Armando Solar-Lezama. Verified lifting of stencil computations. In *ACM SIGPLAN Conf. on Programming Language Design and Implementation*, 2016.

Sean Kandel, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. Wrangler: Interactive visual specification of data transformation scripts. In *SIGCHI Conference on Human Factors in Computing Systems*, pages 3363–3372, 2011.

Nodira Khoussainova, Magdalena Balazinska, Wolfgang Gatterbauer, YongChul Kwon, and Dan Suciu. A case for a collaborative query management system. In *Biennial Conference on Innovative Data Systems Research*, 2009.

Nodira Khoussainova, YongChul Kwon, Magdalena Balazinska, and Dan Suciu. SnipSuggest: Context-aware autocompletion for SQL. *VLDB Endowment*, 4(1):22–33, October 2010.

Bongshin Lee, Rubaiat Habib Kazi, and Greg Smith. SketchStory: Telling more engaging stories with data through freeform sketching. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2416–2425, 2013.

Fei Li and H. V. Jagadish. Constructing an interactive natural language interface for relational databases. *VLDB Endowment*, 8(1):73–84, 2014a.

Fei Li and Hosagrahar V. Jagadish. NaLIR: an interactive natural language interface for querying relational databases. In *ACM SIGMOD International Conference on Management of Data*, pages 709–712, 2014b.

Yunyao Li, Huahai Yang, and H. V. Jagadish. NaLIX: A generic natural language search environment for XML data. *ACM Transactions on Database Systems*, 32(4), 2007.

Guy Lohman. Is Query Optimization a "Solved" Problem? http://wp.sigmod.org/?author=20, 2014. Accessed: 2014-11-22.

Edward Lu and Rastislav Bodík. Quicksilver: Automatic synthesis of relational queries. Technical Report UCB/EECS-2013-68, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, May 2013.

Zohar Manna and Richard Waldinger. Fundamentals of deductive program synthesis. *IEEE Transactions on Software Engineering*, 18(8):674–704, August 1992.

Zohar Manna and Richard J. Waldinger. Deductive synthesis of the unification algorithm. *Science of Computer Programming*, 1(1-2):5–48, 1981.

Microsoft. Entity Framework. http://msdn.microsoft.com/en-us/data/ef.aspx, a. Accessed: 2014-11-22.

Microsoft. Language-integrated query. http://msdn.microsoft.com/en-us/library/bb397926.aspx, b. Accessed: 2014-11-22.

Microsoft. z3 Theorem Prover. http://research.microsoft.com/en-us/um/redmond/projects/z3. Accessed: Apr 5, 2016.

Nathaniel Nystrom, Michael R. Clarkson, and Andrew C. Myers. Polyglot: An extensible compiler framework for Java. In *International Conference on Compiler Construction*, pages 138–152, 2003.

Object Management Group. Common Object Request Broker Architecture (CORBA) Specification, Version 3.3. http://www.omg.org/spec/CORBA/3.3, 2012.

Daniel Perelman, Sumit Gulwani, Dan Grossman, and Peter Provost. Test-driven synthesis. In *ACM SIGPLAN Conf. on Programming Language Design and Implementation*, pages 408–418, 2014.

Phitchaya Mangpo Phothilimthana, Tikhon Jelvis, Rohin Shah, Nishant Totla, Sarah Chasins, and Rastislav Bodik. Chlorophyll: Synthesis-aided compiler for low-power spatial architectures. In *ACM SIGPLAN Conf. on Programming Language Design and Implementation*, pages 396–407, 2014.

Hamid Pirahesh, Joseph M. Hellerstein, and Waqar Hasan. Extensible/rule based query rewrite optimization in Starburst. In *ACM SIGMOD International Conference on Management of Data*, pages 39–48, 1992.

Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. Towards a theory of natural language interfaces to databases. In *International Conference on Intelligent User Interfaces*, pages 149–157, 2003.

Ana-Maria Popescu, Alex Armanasu, Oren Etzioni, David Ko, and Alexander Yates. Modern natural language interfaces to databases: Composing statistical parsing with semantic tractability. In *International Conference on Computational Linguistics*, 2004.

Xiaolei Qian. The deductive synthesis of database transactions. *ACM Transactions on Database Systems*, 18(4):626–677, December 1993.

Lawrence A. Rowe and Kurt A. Shoens. Data abstraction, views and updates in RIGEL. In *ACM SIGMOD International Conference on Management of Data*, pages 71–81, 1979.

Mooly Sagiv, Thomas Reps, and Reinhard Wilhelm. Parametric shape analysis via 3-valued logic. In *ACM Symposium on Principles of Programming Languages*, pages 105–118, 1999.

Sunita Sarawagi, Rakesh Agrawal, and Nimrod Megiddo. Discovery-driven exploration of olap data cubes. In *International Conference on Extending Database Technology*, pages 168–182, 1998.

Anish Das Sarma, Aditya G. Parameswaran, Hector Garcia-Molina, and Jennifer Widom. Synthesizing view definitions from data. In *International Conference on Database Theory*, pages 89–103, 2010.

Eric Schkufza, Rahul Sharma, and Alex Aiken. Stochastic superoptimization. In *International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 305–316, 2013.

Eric Schkufza, Rahul Sharma, and Alex Aiken. Stochastic optimization of floating-point programs with tunable precision. In *ACM SIGPLAN Conf. on Programming Language Design and Implementation*, page 9, 2014.

Joachim W. Schmidt. Some high level language constructs for data of type relation. *ACM Transactions Database Systems*, 2(3):247–261, September 1977.

Praveen Seshadri, Hamid Pirahesh, and T. Y. Cliff Leung. Complex query decorrelation. In *International Conference on Data Engineering*, pages 450–458, 1996.

Burr Settles. Active learning literature survey. Technical Report 1648, Department of Computer Sciences, University of Wisconsin, Madison, 2010.

Rishabh Singh, Sumit Gulwani, and Armando Solar-Lezama. Automated feedback generation for introductory programming assignments. In *ACM SIGPLAN Conf. on Programming Language Design and Implementation*, pages 15–26, 2013.

Armando Solar-Lezama, Liviu Tancau, Rastislav Bodík, Sanjit A. Seshia, and Vijay A. Saraswat. Combinatorial sketching for finite programs. In *International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 404–415, 2006.

Squeryl. Squeryl: A Scala ORM for SQL Databases. http://squeryl.org. Accessed: 2014-11-22.

Saurabh Srivastava and Sumit Gulwani. Program verification using templates over predicate abstraction. In *ACM SIGPLAN Conf. on Programming Language Design and Implementation*, pages 223–234, 2009.

Michael Stonebraker, Daniel Bruckner, Ihab F. Ilyas, George Beskales, Mitch Cherniack, Stanley B. Zdonik, Alexander Pagan, and Shan Xu. Data curation at scale: The data tamer system. In *Biennial Conference on Innovative Data Systems Research*, 2013.

Mike Stonebraker, Daniel J. Abadi, Adam Batkin, Xuedong Chen, Mitch Cherniack, Miguel Ferreira, Edmond Lau, Amerson Lin, Sam Madden, Elizabeth O'Neil, Pat O'Neil, Alex Rasin, Nga Tran, and Stan Zdonik. C-store: A column-oriented DBMS. In *International Conference on Very Large Data Bases*, pages 553–564, 2005.

Singaram Subramanian. How to Identify and Resolve Hibernate N+1 SELECT's Problems. http://architects.dzone.com/articles/how-identify-and-resilve-n1. Accessed: 2015-01-12.

Tableau Software. http://www.tableausoftware.com. Accessed: 2014-11-22.

Emina Torlak and Daniel Jackson. Kodkod: a relational model finder. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 632–647, 2007.

Quoc Trung Tran, Chee-Yong Chan, and Srinivasan Parthasarathy. Query by output. In *ACM SIGMOD International Conference on Management of Data*, pages 535–548, 2009.

Jonathan Traugott. Deductive synthesis of sorting programs. *Journal of Symbolic Computation*, 7(6):533–572, June 1989.

Omer Tripp, Marco Pistoia, Stephen J. Fink, Manu Sridharan, and Omri Weisman. TAJ: effective taint analysis of web applications. In *ACM SIGPLAN Conf. on Programming Language Design and Implementation*, pages 87–97, 2009.

Abhishek Udupa, Arun Raghavan, Jyotirmoy V. Deshmukh, Sela Mador-Haim, Milo M. K. Martin, and Rajeev Alur. TRANSIT: specifying protocols with concolic snippets. In *ACM SIGPLAN Conf. on Programming Language Design and Implementation*, pages 287–296, 2013.

John Whaley and Martin Rinard. Compositional pointer and escape analysis for java programs. In *ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages and Applications*, pages 187–206, 1999.

Ben Wiedermann and William R. Cook. Extracting queries by static analysis of transparent persistence. In *ACM Symposium on Principles of Programming Languages*, pages 199–210, 2007.

Ben Wiedermann, Ali Ibrahim, and William R. Cook. Interprocedural query extraction for transparent persistence. In *ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages and Applications*, pages 19–36, 2008.

Wilos Orchestration Software. http://www.ohloh.net/p/6390. Accessed: Apr 5, 2016.

Yichen Xie and Alex Aiken. Scalable error detection using boolean satisfiability. In *ACM Symposium on Principles of Programming Languages*, pages 351–363, 2005.

Sai Zhang and Yuyin Sun. Automatically synthesizing SQL queries from input-output examples. In *IEEE/ACM International Conference on Automated Software Engineering*, pages 224–234, 2013.

Moshé M. Zloof. Query-by-example: The invocation and definition of tables and forms. In *International Conference on Very Large Data Bases*, pages 1–24, 1975.