ACM.org | Join ACM | About Communications | ACM Resources | Alerts & Feeds

SIGN IN

Indian Institute of Information Technology Guwahati IIITG

Search

HOME | CURRENT ISSUE | NEWS | BLOGS | OPINION | RESEARCH | PRACTICE | CAREERS | ARCHIVE | VIDEOS

Home / Magazine Archive / February 2019 (Vol. 62, No. 2) / February 2019 (Vol. 62, No. 2) / Full Text

RESEARCH HIGHLIGHTS

Technical Perspective: To Do or Not To Do: Extending SQL with Integer Linear Programming?

By Surajit Chaudhuri

Communications of the ACM, February 2019, Vol. 62 No. 2, Page 107
10.1145/3299879

Comments

PRINT

VIEW AS: SHARE:

Relational query languages have enabled the programmer to express queries using a logical model of data without any knowledge of the underlying physical structures. To help applications realize the benefits of such declarative querying of data fully, there has been much work along the following three dimensions:

1. Application programming interfaces (for example, ODBC, JDBC) have been developed to enable applications connect to and access data in a relational database system. However, when connecting using these interfaces, the application programmer must still handle two different programming models. Language integrated query (LINQ) is an elegant example of integration where query expressions are introduced as a first-class citizen in the programming languages to avoid the above problem, and a mapping tool (LINQ to SQL) translates language-integrated queries into SQL for the database backend. More recently, databases have been exposing a REST API for the ease of mobile and web applications.
2. Modern database systems provide extensibility so that applications programmers are not limited to using the built-in types and functions in SQL. All major database systems support user-defined functions that may be used in selection, aggregation, or table expressions in a query. These user-defined functions (potentially with parameters) are written in native SQL or programming languages for which the database server provides runtime support. Such extensibility mechanisms have been used by database systems to add support for data types such as geospatial.
3. The SQL standard has added new operators and constructs to make declarative querying in relational languages more convenient or expressive, for example, recursion, window functions, grouping sets, within group.

Despite the advances that have already taken place along these three dimensions, there continues to be proposals from time to time to further enrich functionality of relational databases to support important classes of applications.

The following paper by Brucato et al. is one such proposal for making relational databases do more. It makes a case for marrying the well-established paradigms of constrained optimization (specifically, ILP or integer linear programming) and traditional SQL querying.

The challenge of augmenting query languages with the power of specifying constraints has been well studied in the literature, both in the context of database querying as well as logic programming. Earlier research has studied schemes for adding constraints on individual rows (beyond simple selection) as well as *aggregate constraints* that the set of answer rows to a query must satisfy collectively. Introduction of aggregate constraints makes query evaluation especially challenging. The paper demonstrates that when you add an optimization criterion to a query language with aggregate constraints to choose among qualifying sets of answer sets, the query evaluation can be accomplished by a combination of the relational query execution engine and an off-the-shelf ILP solver.

The authors explain how such queries may be specified declaratively (referred to as *package queries*). These package queries are evaluated by first executing the traditional relational part of the query and then mapping the constraint satisfaction and objective criterion as an instance of the ILP problem. The extensibility features of the database system, as explained in (b), may be used to add such an ILP solver to the database systems just like the support for user defined functions written in programming languages (for example, Java or C#) other than the native SQL. The paper also addresses techniques for solving large ILP problems using offline partitioning and approximation techniques to break down the global ILP instance into smaller ILP sub-problems. However, while their offline partitioning is a good physical design optimization to have in the repertoire, its applicability also depends on the characteristics of the production workload on the system.

Adding any new functionality to a query language as rich as SQL has complex trade-offs. Issues that influence such a decision are ease of specification of the new functionality in the query, execution efficiency of the enriched query system, data movement, and increased software complexity of the database systems. Moreover, even when a new

SIGN IN for Full Access

User Name

Password

» Forgot Password?

» Create an ACM Web Account

SIGN IN

ARTICLE CONTENTS:

Article

Author

Footnotes

MORE NEWS & OPINIONS

Even a Few Bots Can Shift Public Opinion in Big Ways
The Conversation

Detecting 'DeepFake' Videos in the Blink of an Eye
The Conversation

Governance and Oversight Coming to AI and Automation: Independent Audit of AI Systems
Ryan Carrier

ACM RESOURCES

Paradox 9.0: Introduction
Courses

functionality is incorporated, there is a question of whether the core SQL should be enriched like other examples in (c), as suggested by this paper, or if the functionality should be incorporated strictly via the extensibility mechanisms. Specifically, in this case, an alternative to extending SQL will be to have a separate domain-specific language (potentially using a syntax like that of package queries), interpreted by the ILP solver runtime, and integrated with the database system.

If you are interested in the topic of constraint specification and optimization over data stored in databases, this paper is sure to interest you. Also, it is worth a read for anyone who wants to consider adding extensions to SQL to ease application tasks, as the authors illustrate the key dimensions of what it takes to add any new functionality to relational querying: language extension, changes to the query execution engine, and techniques to cope with scale.

Back to Top

Author

Surajit Chaudhuri is a Distinguished Scientist at Microsoft Research, Redmond, WA, USA.

Back to Top

Footnotes

To view the accompanying paper, visit doi.acm.org/10.1145/3299881

Copyright held by author.
Request permission to (re)publish from the owner/author

The Digital Library is published by the Association for Computing Machinery. Copyright © 2019 ACM, Inc.

No entries found

Comment on this article

Signed comments submitted to this site are moderated and will appear if they are relevant to the topic and not abusive. Your comment will appear with your username if published. View our policy on comments

(Please sign in or create an ACM Web Account to access this feature.)

Create an Account

SUBMIT FOR REVIEW

Copyright © 2019 by the ACM. All rights reserved.