Volume 9, Special Issue No. 1, February 2018

ISBN: 978-93-5300-455-2

International Journal of Advanced Research in Computer Science
(ISSN: 0976-5697)
RESEARCH PAPER

ICRACC 2018
(23rd–24th February 2018)

Available Online at www.ijarcs.info

ANSWERING PATTERN QUERIES USING VIEWS

Dr.S.T.Deepa
Associate professor, Department of Computer Science
Shri S.S.Shasun Jain College for Women
Chennai, India
email: deepatheodore@gmail.com

Ms.G.S. Shailaja
Computer Instructor
Chennai Higher Secondary School
Chennai, India
e-mail: shakthi.s2012@gmail.com

ABSTRACT

Information is playing an important role in our lives. One of the major sources of information is databases. Databases and database technology are having major impact on the growing use of computers. In order to retrieve information from a database, one needs to formulate a query in such way that the computer will understand and produce the desired output. The Structured Query Language (SQL) norms are been pursued in almost all languages for relational database systems. However, not everybody is able to write SQL queries as they may not be aware of the structure of the database. So there is a need for non-expert users to query relational databases in their natural language instead of working with the values of the attributes. The idea of using natural language instead of SQL, has promoted the development of Natural Language Interface to Database systems (NLIDB). The need of NLIDB is increasing day by day as more and more people access information through web browsers, PDA's and cell phones. In this paper we introduce an intelligent interface for database. We prove that our NLIDB is guaranteed to map a natural language query to the corresponding SQL query. We have tested our system on Northwind database and show that our NLIDB compares favourably with MS English Query product

Keywords: Natural language, pattern, containment, item set, queries.

I. INTRODUCTION

Data mining is the process of discovering interesting patterns from massive amounts of data. As a knowledge discovery process, it typically involves data cleaning, data integration, data selection, data transformation, pattern discovery, pattern evaluation, and knowledge presentation. The major dimensions of data mining are data, knowledge, technologies, and applications. The book focuses on fundamental data mining concepts and techniques for discovering interesting patterns from data in various applications. Prominent techniques for developing effective, efficient, and scalable data mining tools are focused on. This chapter discusses why data mining is in high demand and how it is part of the natural evolution of information technology. It defines data mining with respect to the knowledge discovery process. Next, data mining from many aspects, such as the kinds of data that can be mined, the kinds of knowledge to be mined, the kinds of technologies to be used and targeted applications are discussed which helps gain a multidimensional view of data mining. Data mining can be conducted on any kind of data as long as the data are meaningful for a target application, such as database data, data warehouse data, transactional data, and advanced data types. Finally major data mining research and development issues are outlined.

II. LITERATURE REVIEW

The problem of answering queries using views is to find efficient methods of answering a query using a set of previously defined materialized views over the database, rather than accessing the database relations[1]. The problem has recently received significant attention because of its relevance to a wide variety of data management problems. In query optimization, finding a rewriting of a query using a set of materialized views can yield a more efficient query execution plan. To support the separation of the logical and physical views of data, a storage schema can be described using views over the logical schema. As a result, finding a query execution plan that accesses the storage amounts to solving the problem of answering queries using views. Finally, the problem arises in data integration systems, where data sources can be described as precomputed views over a mediated schema. This article surveys the state of the art on the problem of answering queries using views, and synthesizes the disparate works into a coherent framework. We describe the different applications of the problem, the algorithms proposed to solve it and the relevant theoretical results.

Data integration is the problem of combining data residing at different sources, and providing the user with a unified view of these data[2]. The problem of designing data integration systems is important in current real world applications, and is characterized by a number of issues that are interesting from a theoretical point of view. This document presents on overview of the material to be presented in a tutorial on data integration. The tutorial is focused on some of the theoretical issues that are relevant for data integration. Special attention will be devoted to the following aspects: modeling a data integration application, processing queries in data integration, dealing with inconsistent data sources, and reasoning on queries.

The problem of answering tree pattern queries using views was revisited[3].They first show that, for queries and views that do not have nodes labeled with the wildcard *, there is an alternative to the approach of query rewriting which does not require us to find any rewritings explicitly yet which produces the same answers as the maximal contained rewriting. Then, using the new approach, they give a simple criterion and a corresponding algorithm for identifying redundant view answers, which are view answers that can be ignored when evaluating the maximal contained rewriting. Finally, for queries and views that do have nodes labeled *,they provide a method to find the maximal contained rewriting and show how to answer the query using views without explicitly finding the rewritings.

View-based query processing requires answering a query posed to a database only on the basis of the information on a set of views, which are again queries over the same database[4].This problem is relevant in many aspects of database management, and has been addressed by means of two basic approaches: query rewriting and query answering. In the former approach, one tries to compute a rewriting of the query in terms of the views, whereas in the latter, one aims at directly answering the query based on the view extensions. They study view based query processing for the case of regular-path queries, which are the basic querying mechanisms for the emergent field of semi structured data. Based on recent results, They first show that a rewriting is in general a co-NP function wrt to the size of view extensions. Hence, the problem arises of characterizing which instances of the problem admit a rewriting that is PTIME. A second contribution of the work is to establish a tight connection between view based query answering and constraint satisfaction problems, which allows us to show that the above characterization is going to be difficult. As a third contribution, we present two methods for computing PTIME rewritings of specific forms. The first method, which is based on the established connection with constraint

satisfaction problems, gives us rewritings expressed in Data log with a fixed number of variables. The second method, based on automata-theoretic techniques, gives us rewritings that are formulated as unions of conjunctive regular-path queries with a fixed number of variables.

The problem of query rewriting for TSL, a language for querying semi structured data is addressed [5]. They develop and present an algorithm that, given a semi structured query q and a set of semi structured views V, finds rewriting queries, i.e., queries that access the views and produce the same result as q. The algorithm is based on appropriately generalizing containment mappings, the chase, and unication techniques that were developed for structured, relational data. We also develop an algorithm for equivalence checking of TSL queries. We show that the algorithm is sound and complete for TSL, i.e., it always finds every TSL rewriting query of q, and we discuss its complexity. They extend the rewriting algorithm to use available structural constraints (such as DTDs) to find more opportunities for query rewriting. We currently incorporate the algorithm in the TSIMMIS system.

The problem of maintaining materialized views of graph structured data was studied **[6]**. The base data consists of records containing identifiers of other records. The data could represent traditional objects (with methods, attributes, and a class hierarchy), but it could also represent a lower level data structure. They define simple views and materialized views for such graph structured data, analyzing options for representing record identity and references in the view. We develop incremental maintenance algorithms for these views.

[7]Developers of rapidly growing applications must be able to anticipate potential scalability problems before they cause performance issues in production environments. A new type of data independence, called scale independence, seeks to address this challenge by guaranteeing a bounded amount of work is required to execute all queries in an application, independent of the size of the underlying data. While optimization strategies have been developed to provide these guarantees for the class of queries that are scale-independent when executed using simple indexes, there are important queries for which such techniques are insufficient. Executing these more complex queries scale-independently requires precomputation using incrementally-maintained materialized views. However, since this precomputation effectively shifts some of the query processing burden from execution time to insertion time, a scale-independent system must be careful to ensure that storage and maintenance costs do not threaten scalability. In this paper, we describe a scale independent view selection and maintenance system, which uses novel static analysis techniques that ensure that created views do not themselves become scaling bottlenecks. Finally, we present an empirical analysis that includes all the queries from the TPC-W benchmark and validates our implementation's ability to maintain nearly constant high quantile query and update latency even as an application scales to hundreds of machines.

[8]To make query answering feasible in big datasets, practitioners have been looking into the notion of scale independence of queries. Intuitively, such queries require only a relatively small subset of the data, whose size is determined by the query and access methods rather than the size of the dataset itself. This paper aims to formalize this notion and study its properties. We start by defining what it means to be scale-independent, and provide matching upper and lower bounds for checking scale independence, for queries in various languages, and for combined and data complexity. Since the complexity turns out to be rather high, and since scale-independent queries cannot be captured syntactically, we develop sufficient conditions for scale independence. We formulate them based on access schemas, which combine indexing and constraints together with bounds on the sizes of retrieved data sets. We then study two variations of scale independent query answering, inspired by existing practical systems. One concerns incremental query answering: we check when query answers can be maintained in response to updates scale-independently. The other explores scale independent query rewriting using views.

The design and the evaluation of the ADMS optimizer was described**[9]**. Capitalizing on a structure called Logical Access Path Schema to model the derivation relationship among cached query results, the optimizer is able to perform query matching coincidently with the optimization and generate more efficient query plans using cached results. The optimizer also features data caching and pointer caching, different cache replacement strategies, and different cache update strategies. An extensive set of experiments were conducted and the results showed that pointer caching and dynamic cache update strategies substantially speedup query computations and, thus, increase query throughput under situations with fair query correlation and update load. The requirement of the cache space is relatively small and the extra computation overhead introduced by the caching and matching mechanism is more than offset by the time saved in query processing.

The prevalent use of XML highlights the need for a generic, flexible access-control mechanism for XML documents that supports efficient and secure query access, without revealing sensitive information to unauthorized users 10]. A novel paradigm for specifying XML security constraints and investigates the enforcement of such constraints during XML query evaluation is introduced. The approach is based on the novel concept of security views, which provide for each user group (a) an XML view consisting of all and only the information that the users are authorized to access, and (b) a view DTD that the XML view conforms to. Security views effectively protect sensitive data from access and potential inferences by unauthorized users, and provide authorized users with necessary schema information to facilitate effective query formulation and optimization. We propose an efficient algorithm for deriving security view definitions from security policies (defined on the original document DTD) for different user groups. We also develop novel algorithms for XPath query rewriting and optimization such that queries over security views can be efficiently answered without materializing the views. Our algorithms transform a query over a security view to an equivalent query over the original document, and effectively prune query nodes by exploiting the structural properties of the document DTD in conjunction with approximate XPath containment tests. Our work is the first to study a flexible, DTD-based access-control model for XML and its implications on the XML query-execution engine. Furthermore, it is among the first efforts for query rewriting and optimization in the presence of general DTDs for a rich class of XPath queries. An empirical study based on real-life DTDs verifies the effectiveness of our approach.

[11] Graph pattern matching is typically defined in terms of subgraph isomorphism, which makes it an np-complete problem. Moreover, it requires bijective functions, which are often too restrictive to characterize patterns in emerging applications. They proposed a class of graph patterns, in which an edge denotes the connectivity in a data graph within a predefined number of hops. In addition, we define matching based on a notion of bounded simulation, an extension of graph simulation. We show that with this revision, graph pattern matching can be performed in cubic-time, by providing such an algorithm. We also develop algorithms for incrementally finding matches when data graphs are updated, with performance guarantees for dag patterns. We experimentally verify that these algorithms scale well, and that the revised notion of graph pattern matching allows us to identify communities commonly found in real-world networks.

[12]They presented algorithms for computing similarity relations of labeled graphs. Similarity relations have applications for the refinement and verification of reactive systems. For finite graphs, we present an O(mn) algorithm for computing the similarity relation of a graph with n vertices and m edges (assuming m n). For effectively presented infinite graphs, we present a symbolic similarity-checking procedure that terminates if a finite similarity relation exists. We show that 2D rectangular automata, which model discrete reactive systems with continuous environments, define effectively presented infinite graphs with finite similarity relations. It follows that the refinement problem and the 8CTL model-checking problem are decidable for 2D rectangular automata.

Describing social positions and roles is an important topic within social network analysis. One approach is to compute a suitable equivalence relation on the nodes of the target network [13]. One relation that is often used for this purpose is regular equivalence, or bisimulation, as it is known within the field of computer science. In this paper we consider a relation from computer science called simulation relation. Simulation creates a partial order on the set of actors in a network and we can use this order to identify actors that have characteristic properties. The simulation relation can also be used to compute simulation equivalence which is a less restrictive equivalence relation than regular equivalence but is still computable in polynomial time. This paper primarily considers weighted directed networks and we present definitions of both weighted simulation equivalence and weighted regular equivalence. Weighted networks can be used to model a number of network domains, including information flow, trust propagation, and communication channels. Many of these domains have applications within homeland security and in the military, where one wants to survey and elicit key roles within an organization. Identifying social positions can be difficult when the target organization lacks a formal structure or is partially hidden.

III. METHODLOGY

Determining Pattern Containment

To do this, we first propose a sufficient and necessary condition to characterize pattern containment. We then develop a cubic time algorithm based on the characterization. Sufficient and necessary condition. To characterize pattern containment, we introduce a notion of view matches. Consider a pattern query Qs and a set V of view definitions. For each V 2 V, let VðQsÞ ¼ fðeV; SeVÞ j eV 2 Vg, by treating Qs as a data graph. Obviously, if V E Qs, then SeV is the nonempty match set of eV for each edge eV. We define the view match from V to Qs, denoted by MQsV , to be the union of SeV for all eV in V.

Minimal Containment Problem:

Given a pattern query Qs and a set V of view definitions,it returns either a nonempty subset V0 of V that minimally contains Qs, or ; to indicate that Qs 6v V.

Algorithm minimal initializes (1) an empty set V0 for selected views, (2) an empty set S for view matches of V0,and (3) an empty set E for edges in view matches. It also maintains an index M that maps each edge e in Qs to a set of views. Similar to algorithm contain, minimal first computes MQsVi for all Vi 2 V. In contrast to contain that simply merges the view matches, it extends S with a new view match MQsVi only if MQsVi contains a new edge not in E, and updates M accordingly. The for loop stops as soon as E ¼ Ep, as Qs is already contained in V0. If E 6¼ Ep after the loop, it returns, since Qs is not contained is V . The algorithm then eliminates redundant views, by checking whether the removal of Vj causes MðeÞ¼ ; for some e2 MQsVj. If no such e exists, it removes Vj from V0. After all view matches are checked, minimalreturns V0.

The algorithm is denoted as minimum. Given a pattern Qs and a set V of views,

minimum identifies a subset V0 of V such that (1) Qs v V0 ifQs v V and (2) cardðV0Þ _ log ðjEpjÞ _ cardðVOPTÞ, whereVOPT is a minimum subset of V that contains Qs. That is, the approximation ratio of minimum is OðlogjEpjÞ, where jEpj is typically small. The algorithm iteratively finds the “top” view whose

view match can cover most edges in Qs that are not yet covered.To do this, we define a metric aðVÞ for a view V, whereaðVÞ ¼ jMQsV n EcjjEpj:Here Ec is the set of edges in Ep that have been covered byselected view matches, and aðV Þ indicates the amount ofuncovered edges that MQsV covers. We select V with the largesta in each iteration, and maintain a accordingly. Similar to minimal, algorithm minimum computes the view match MQsVi for each Vi 2 V, and collects them in a set S. It then does the following. (1) It selects view Viwith the largest a, and removes MQsVifrom S. (2) Itmerges Ec with MQsViif MQsVicontains some edges that arenot in Ec, and extends V0 with Vi. During theloop, if Ec equals Ep, the set V0 is returned. Otherwise,minimum returns indicating that Qs 6v V.

Maximally Contained Rewriting

A pattern query Qs0 is a subquery of Qs, denoted as Qs0 _ Qs, if it is an edge induced subgraph of Qs, i.e., Qs0 is a subgraph of Qs consisting of a subset of edges of Qs, together with their endpoints as the set of nodes. Query Qs0 is called a contained rewriting of Qs using a set V of view definitions if_ Qs0 _ Qs, i.e., Qs0 is a subquery of Qs, and_ Qs0 v V, i.e., Qs0 can be answered using V. Such a rewriting Qs0 is a maximally contained rewriting ofQs using V if there exists no contained rewriting Qs00 such that Qs0 Qs00, i.e., there exists no larger contained rewritingQs00 with more edges than Qs0.Query-driven approximation scheme. When Qs is not contained in V, we can still efficiently answer Qs in a (possibly

big) graph G following two approaches. (1) One may firs tidentify a maximally contained rewriting Qs0 of Qs using V,and then compute Qs0ðGÞ as approximate answers to Qs, bysimply invoking the algorithm MatchJoin. (2) Alternatively, one may compute exact answers QsðGÞ by using Qs0ðGÞ and by accessing a small fraction GQs of G, such thatQsðGÞ ¼ Qs0ðGÞ [fðGQs Þ. Here fðGQs Þ first locates thematches of Qs0ðGÞ in the original graph G and then verifies the matches for Qs by visiting neighborhood of those matches, a small number of nodes and edges in G that constitute GQs ; this is the approach suggested, referred

to as scale-independent query answering using views there. Due to the space constraint, we focus on approximate answersQs0ðGÞ in this paper. That is, when limited views are available, we can still approximately answer pattern queries in

big graphs by relaxing Qs to maximally contained rewritingQs0, using those views. Accuracy. Given a graph G, we measure the quality of the approximate answers Qs0ðGÞ versus the true matches in the exact answers QsðGÞby following the F-measure:

$$Acc = 2 * \frac{recall * precision}{recall + precision};$$

where recall = #true matches found #true matches , and precision =#true matches found # matches . Here #matches is the number of all (edge)matches found by Qs0ðGÞ using views, #true matches is the number of all matches in QsðGÞ; and #true matches found is the number of all the true matches in both Qs0ðGÞ and QsðGÞ. Intuitively, a high precision means that many matches inQs0ðGÞ are true matches, and a high recall means Qs0ðGÞcontains most of the true matches in QsðGÞ. The larger Acc that can be induced by Qs0, the better. If Qs0 is equivalent toQs, i.e., Qs0ðGÞ = QsðGÞ for all G, Acc takes the maximum value 1:0. Observe that for any edge e in Qs, if e is covered by Qs0, then for any G, the match set Se of e in QsðGÞ is asubset of the match set S0eof e in Qs0ðGÞ; that is, Qs0ðGÞ finds all candidate matches of e in G.

IV. RESULTS AND DISCUSSIONS

The criteria for comparing the methods of XML queries and graph pattern queries

- **Accuracy** − Accuracy of classifier refers to the ability of classifier. It predict the class label correctly and the accuracy of the predictor refers to how well a given predictor can guess the value of predicted attribute for a new data.

- **Speed** − This refers to the computational cost in generating and using the classifier or predictor.

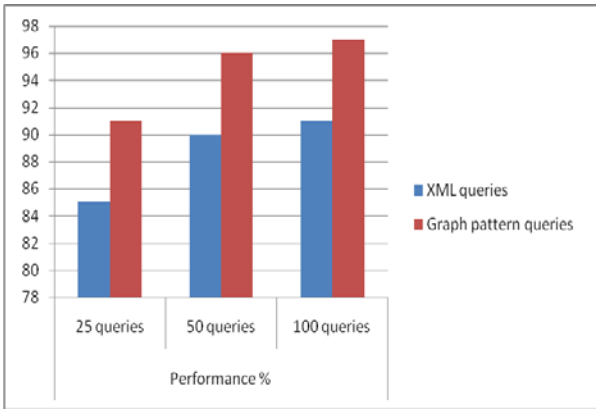
- **Robustness** − It refers to the ability of classifier or predictor to make correct predictions from given noisy data.

- **Scalability** − Scalability refers to the ability to construct the classifier or predictor efficiently; given large amount of data.

Interpretability − It refers to what extent the classifier or predictor understands

The performance of the both the queries are analysed.

Method	Performance %		
	25 queries	50 queries	100 queries
XML queries	85	90	91
Graph pattern queries	91	96	97



V. CONCLUSION

We have proposed a notion of pattern containment to characterize what pattern queries can be answered using views, and provided such an efficient matching algorithm. We have also identified three fundamental problems for pattern containment, established their complexity, and developed effective (approximation)algorithms. When a pattern query is not contained in available views, we have developed efficient algorithms for computing maximally contained rewriting using views to get approximate answers. Our experimental results have verified the efficiency and effectiveness of our techniques. These results extend the study of query answering using views from relational and XML queries to graph pattern queries. Finally, to find a practical method to query "big" social data, one needs to combine techniques such as view-based, distributed, incremental, and compression methods. The efficiency and effectiveness of the technique are verified through experimental results. The study of query answering is extended from These results extend the study of query relational to graph pattern queries.

VI. REFERENCES

[1] A. Y. Halevy, "Answering queries using views: A survey," VLDBJ., vol. 10, no. 4, pp. 270–294, 2001.

[2] M. Lenzerini, "Data integration: A theoretical perspective," inProc. 21st ACM SIGMOD-SIGACT-SIGART Symp. Principles DatabaseSyst., 2002, pp. 233–246.

[3] L. V. S. Lakshmanan, W. H. Wang, and Z. J. Zhao, "Answeringtree pattern queries using views," in Proc. 32nd Int. Conf. VeryLarge Data Bases, 2006, pp. 571–582.

[4] J. Wang, J. Li, and J. X. Yu, "Answering tree pattern queries usingviews: A revisit," in Proc. 14th Int. Conf. Extending Database Technol.,2011, pp. 153–164.

[5] X. Wu, D. Theodoratos, and W. H. Wang, "Answering XMLqueries using materialized views revisited," in Proc. 18th ACMConf. Inf. Knowl. Manag., 2009, pp. 475–484.

[6] D. Calvanese, G. D. Giacomo, M. Lenzerini, and M. Y. Vardi,"View-based query processing and constraint satisfaction,"in Proc. 15th Annu. IEEE Symp. Logic Comput. Sci., 2000, p. 361.

[7] Y. Papakonstantinou and V. Vassalos, "Query rewriting for semistructureddata," in Proc. ACM SIGMOD Int. Conf. Manag. Data,1999, pp. 455–466.

[8] Y. Zhuge and H. Garcia-Molina, "Graph structured views andtheir incremental maintenance," in Proc. 14th Int. Conf. Data Eng.,1998, pp. 116–125.

[9] M. Armbrust, E. Liang, T. Kraska, A. Fox, M. J. Franklin, and D. A.Patterson, "Generalized scale independence through incrementalprecomputation," in Proc. ACM SIGMOD Int. Conf. Manag. Data,2013, pp. 625–636.

[10] W. Fan, F. Geerts, and L. Libkin, "On scale independence for queryingbig data," in Proc. 33rd ACM SIGMOD-SIGACT-SIGARTSymp. Principles Database Syst., 2014, pp. 51–62.

[11] C.-M. Chen and N. Roussopoulos, "The implementation and performanceevaluation of the ADMS query optimizer: Integratingquery result caching and matching," in Proc. 4th Int. Conf. ExtendingDatabase Technol.: Adv. Database Technol., 1994, pp. 323–336.

[12] W. Fan, C. Y. Chan, and M. N. Garofalakis, "Secure XML queryingwith security views," in Proc. ACM SIGMOD Int. Conf. Manag.Data, 2004, pp. 587–598.

[13] W. Fan, J. Li, S. Ma, N. Tang, Y. Wu, and Y. Wu, "Graph patternmatching: From intractable to polynomial time," Proc. VLDBEndowment, vol. 3, no. 1, pp. 264–265, 2010.