International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2017 IJSRCSEIT | Volume 2 | Issue 2 | ISSN : 2456-3307

Website Content Management System

B. Sankari[1], S. Ajikumar[2]

[1]PG Scholar, Department of M.Sc (Software Engineering), PSN College of Engineering & Technology, Tirunelveli, Tamilnadu, India

[2] Research Supervisor, Department of M.Sc (Software Engineering), PSN College of Engineering & Technology, Tirunelveli, Tamilnadu, India

ABSTRACT

Bug Tracking, Help Desk Ticketing, issue raising, search facility, help information, issue resolution. Issues related to software projects can be raised, tracked and resolved by Employees of different departments. Resolved issues can be allowed to access from Knowledge Base as Knowledge elements. The different groups and representatives can interact each other through emails.The issue tracking system does all the jobs that are done in conventional system but ,here , everything is done in more formal and efficient manner. All the users of organization can interact with each other through the Issue Tracking System. This system acts as an interface between the employees thereby enabling them to forward their issues to the centralized Issue tracking system. Hence, making the work easy for both the issue raiser and the resolved. It totally avoids the involvement of middlemen in getting resolution for a particular issue.The Issue Tracking system is an intranet application, which provides information about issues in software projects, in detail. This product develops a system that can be used by all the departments of a software organization. In the conventional method, all the issues are dealt manually .The progress of the issues are also checked in person, which is a tedious task. Here, in Issue Tracking, it fulfills different requirements of administrator and employees of a software development organization efficiently. The specific purpose of the system is to gather and resolve issues that arise in different projects handled by the organization.

Keywords : Network, data, Local area Network, Transmission control protocol, Internet Protocol.

I. INTRODUCTION

Telecommunications network which allows computers to exchange data. In computer networks, networked computing devices exchange data with each other using a data link. The connections between nodes are established using either cable media or wireless media. The best-known computer network is the Internet.

Network computer devices that originate, route and terminate the data are called network nodes. Nodes can include hosts such as personal computers, phones, servers as well as networking hardware. Two such devices can be said to be networked together when one device is able to exchange information with the other device, whether or not they have a direct connection to each other .Computer networks differ in the transmission medium used to carry their signals, communications protocols to organize network traffic, the network's size, topology and organizational intent.

Computer networks support an enormous number of applications and services such as access to the World Wide Web, digital video, digital audio, shared use of application server, printers, and fax machines, and use of email and instant messagingapplications as well as many others. In most cases, application-specific communications protocols are layered (i.e. carried as payload) over other more general communications protocols.

A computer network facilitates interpersonal communications allowing users to communicate efficiently and easily via various means: email, instant messaging, chat rooms, telephone, video telephone calls, and video conferencing. Providing access to informationon shared storage devices is an important feature of many networks. A network allows sharing of files, data, and other types of information giving authorized users the ability to access information stored on other computers on the network. A network allows sharing of network and computing resources. Users

may access and use resources provided by devices on the network, such as printing a document on a shared network printer. Distributed computing uses computing resources across a network to accomplish tasks.

A computernetwork may be usedby computer crackers to deploy computer viruses or computer worm son devices connected the network, or to prevent these devices from accessing the network via a denial of service attack.

Computer communication links that do not support packets, such as traditionalpoint-to-point telecommunication links, simply transmit data as a bit stream. However, most information in computer networks is carried in packets. A network packet is a formatted unit of data (a list of bits or bytes, usually a few tens of bytes to a few kilobytes Long) carried by a packet-switched network. In packet networks, the data is formatted into packets that are sent through the network to their destination. Once the packets arrive they are reassembled into their original message. With packets, the band width of the transmission medium can be better shared among users than if the network were circuit switched. When one user is not sending packets, the link can be filled with packets from others users, and so the cost can be shared, with relatively little interference, provided the link isn't over used. Packets consist of two kinds of data: control information, and user data (payload). The control information provides data the network needs to deliver the user data, for example: source and destination network addresses, error detection codes, and sequencing information.

User queries. For both query options, the query and the preferences are given as input to the query parser. Apart from the core PrefDB query processing strategies that blend preference evaluation into query processing, we have also implemented a set of plug-in methods, which are described in the Appendix. Below is an overview of the core PrefDB modules

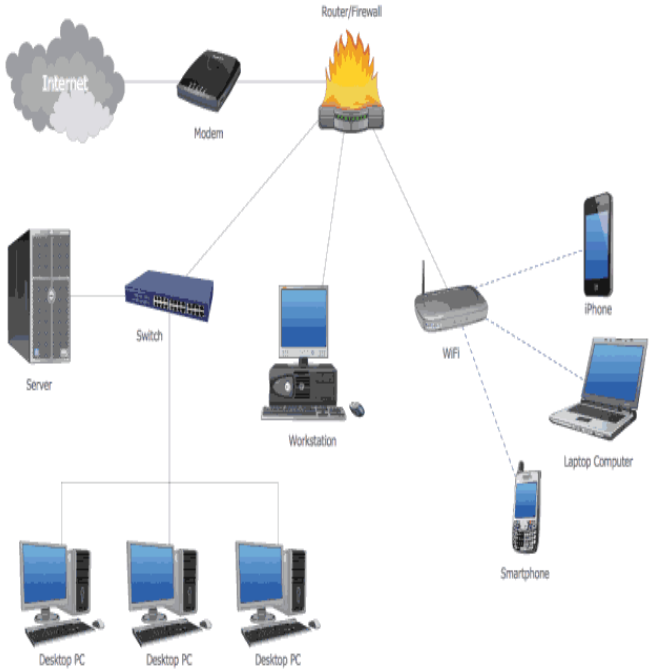


Figure 2. System Architecture

- The profile manager selects from the database preferences that can be combined with the conditions of the issued query. For this purpose, we use the preference selection algorithm proposed in [20]
- The query parser takes as input the query and preferences and generates an extended query plan that is passed to the PrefDB query optimizer.
- The query optimizer improves the input plan by applying a set of algebraic rules. This improved plan and a cost model for preference evaluation are used for generating alternative plans that interleave preference evaluation and query processing in different ways and for picking the plan with the cheapest estimated cost.
- The execution engine realizes the execution of the query plan selected by the query optimizer using one of our execution methods. We discuss

II. METHODS AND MATERIAL

A. Related Work

The concept of preference-aware query processing appears in many applications, where there is a matter of choice among alternatives, including query personalization [10], [18], [20], recommendations [4] and multi-criteria decision making [9], [13]. We discuss prior work with respect to how preferences are represented in the context of relational data and how

they are integrated and processed in queries. In representing preferences, there are two approaches. In the qualitative approach, preferences are specified using binary predicates called preference relations [5], [10], [18]. In quantitative approaches, preferences are expressed as scores assigned to tuples [6], [23] be specified based on any combination of scores, confidences and context. Our framework allows us to process in a uniform way all these different query and preference types. In terms of preference integration and processing, one approach is to translate preferences into conventional queries and execute them over the DBMS [14], [19], [20], [21], [24]. Several efficient algorithms have been proposed for processing different types of queries, including top-k queries [13] and skylines [9]. These algorithms as well as query translation methods are typically implemented outside the DBMS. Thus, they can only apply coarse grained query optimizations, such as reducing the number of queries sent to the DBMS. Further, as we will also demonstrate experimentally plug-in methods do not scale well when faced with multi-join queries or queries involving many preferences. Native implementations modify the database engine by adding specific physical operators and algorithms. RankSQL [23] extends the relational algebra with a new operator called rank that enables pipelining and hence optimizing top-k queries. Another example of operator is the winnow operator [10], which selects all tuples corresponding to the Pareto optimal set. Our approach is different from existing works in several ways. First, existing techniques are limited to a particular type of query. In contrast to these approaches, we consider preference evaluation (how preferences are evaluated on data) and selection of the preferred tuples that will comprise the query answer as two operations. We focus on preference evaluation as a single operator that can be combined with other operators and we use its algebraic properties in order to develop generic query optimization and processing techniques. Finally, we follow a hybrid implementation that is closer to the database than plug-in approaches yet not purely native, thus combining the pros of both worlds. A different approach to flexible processing of queries with preferences is enabled in FlexPref [22]. FlexPref allows integrating different preference algorithms into the database with minimal changes in the database engine by simply defining rules that determine the most preferred tuples. Once these rules are specified a new operator can be used inside queries. It is worth noting that both FlexPref and our work are motivated by the limitations of plug-in and native approaches. FlexPref approaches the problem from an extensibility viewpoint. Our focus is on the problem of preference evaluation as an operator that is separate from the selection of preferred answers, and we study how this operator can be integrated into query processing in an effective yet not obtrusive to the database engine way.

B. Proposed Methodology

In this paper, we first construct an extended query plan that contains all operators that comprise a query and we optimize it. Then, for processing the optimized query plan, our general strategy is to blend query execution with preference evaluation and leverage the native query engine to process parts of the query that do not involve a prefer operator. Given a query with preferences, the goal of query optimization is to minimize the cost related with preference evaluation. Based on the algebraic properties of prefer, we apply a set of heuristic rules aiming to minimize the number of tuples that are given as input to the prefer operators. We further provide a cost-based query optimization approach. Using the output plan of the first step as a skeleton and a cost model for preference evaluation, the query optimizer calculates the costs of alternative plans that interleave preference evaluation and query processing in different ways. Two plan enumeration methods, i.e., a dynamic programming and a greedy algorithm are proposed. For executing an optimized query plan with preferences, we describe an improved version of our processing algorithm (GBU) (an earlier version is described in. The improved algorithm uses the native query engine in a more efficient way by better grouping operators together and by reducing the out-of-the-engine query processing.

Modules:

Registration & Interest Sum up
Query Formation
Query Optimization & Execution

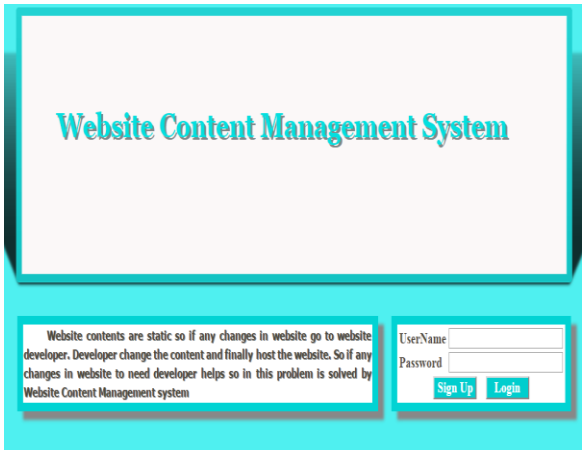
A preferential query combines p-relations, extended relational and prefer operators and returns a set of tuples that satisfy the boolean query conditions along with their score and confidence values that have been calculated after evaluating all prefer operators on the corresponding relations. Intuitively, the better a tuple matches preferences and the more (or more confident)

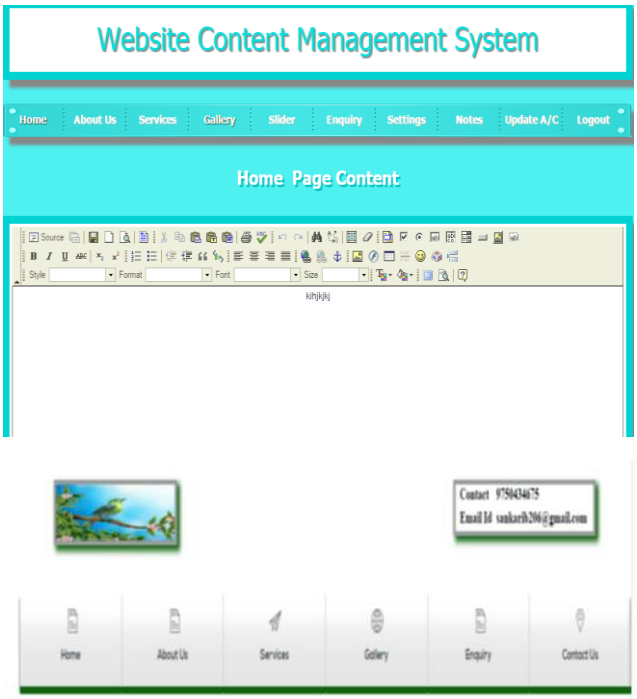
preferences it satisfies, the higher its final score and confidence will be, respectively. The query parser adds a prefer operator for each preference. Finally, the query parser checks for each preference, whether it involves an attribute (either in the conditional or the scoring part) that does not appear in the query and modifies project operators, such that these attributes will be projected as well.

Proportional to the number of tuples flowing through the operators in the query plan. Assuming a fixed position for the other operators, the goal of our query optimizer is essentially to place the prefer operators inside the plan, such that the number of tuples flowing through the score tables is minimized. The execution engine of PrefDB is responsible for processing a preferential query and supports various algorithms.

III. RESULTS AND DISCUSSION

The implementation results can be shown as figure below





IV. CONCLUSION

Various web based application makes the human work schedule as a simple one, as well as in this web based project tracking and resolving the clients send their queries to the developer of the software via online. The people who are going to access in this software project are the system administrator, staff, and user.

The administrator controls the number of modules and the administrator maintains user queries and everything. The person who develop this project and who answer various queries of the user is the staff. The users are the persons who are use the software and asking query to developer.

The tracking and resolving is mechanism of online approach of the customer satisfaction using the ASP.Net programming.

V. REFERENCES

[1]. DBLP computer science bibliography. http://dblp.uni-trier.de/.

[2]. IMDB movie database. http://www.imdb.com.

[3]. Query templates. http://tinyurl.com/8zs3e77.

[4]. G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. TKDE, 17(6):734–749, 2005.

[5]. R. Agrawal, R. Rantzau, and E. Terzi. Context-sensitive ranking. In SIGMOD, pages 383–394, 2006.
[6]. R. Agrawal and E. L. Wimmers. A framework for expressing and combining preferences. In SIGMOD, pages 297–306, 2000.
[7]. A. Arvanitis and G. Koutrika. PrefDB: Bringing preferences closer to the DBMS. In SIGMOD, pages 665–668, 2012.
[8]. A. Arvanitis and G. Koutrika. Towards preference-aware relational databases. In ICDE, pages 426–437, 2012.
[9]. S. Borzs ¨ onyi, D. Kossmann, and K. Stocker. The skyline operator. ¨ In ICDE, pages 421–430, 2001.
[10]. J. Chomicki. Preference formulas in relational queries. TODS, 28(4):427–466, 2003.
[11]. V. Christophides, D. Plexousakis, M. Scholl, and S. Tourtounis. On labeling schemes for the semantic web. In WWW, pages 544–555, 2003.
[12]. W. W. Cohen, R. E. Schapire, and Y. Singer. Learning to order things. J. Artif. Intell. Res. (JAIR), 10:243–270, 1999.
[13]. R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. In PODS, pages 102–113, 2001.
[14]. P. Georgiadis, I. Kapantaidakis, V. Christophides, E. M. Nguer, and N. Spyratos. Efficient rewriting algorithms for preference queries. In ICDE, pages 1101–1110, 2008.
[15]. S. Holland, M. Ester, and W. Kießling. Preference mining: A novel approach on mining user preferences for personalized applications. In PKDD, pages 204–216, 2003.
[16]. I. F. Ilyas, W. G. Aref, and A. K. Elmagarmid. Supporting top-k join queries in relational databases. In VLDB, pages 754–765, 2003.
[17]. T. Joachims. Optimizing search engines using clickthrough data. In KDD, pages 133–142, 2002.
[18]. W. Kießling. Foundations of preferences in database systems. In VLDB, pages 311–322, 2002.
[19]. W. Kießling and G. Kostler. Preference SQL - design, implementation, experiences. In VLDB, pages 990–1001, 2002.
[20]. G. Koutrika and Y. E. Ioannidis. Personalization of queries in database systems. In ICDE, pages 597–608, 2004.
[21]. M. Lacroix and P. Lavency. Preferences: Putting more knowledge into queries. In VLDB, pages 217–225, 1987.
[22]. J. Levandoski, M. Mokbel, and M. Khalefa. FlexPref: A framework for extensible preference evaluation in database systems. In ICDE, pages 828–839, 2010.
[23]. C. Li, K. C.-C. Chang, I. F. Ilyas, and S. Song. RankSQL: Query algebra and optimization for relational top-k queries. In SIGMOD, pages 131–142, 2005.
[24]. C. Mishra and N. Koudas. Stretch 'n' shrink: Resizing queries to user preferences. In SIGMOD, pages 1227–1230, 2008.
[25]. P. G. Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price. Access path selection in a relational database management system. In SIGMOD, pages 23–34, 1979.
[26]. K. Stefanidis, E. Pitoura, and P. Vassiliadis. Adding context to preferences. In ICDE, pages 846–855, 2007.