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

Internet of Things for Health Care

M. Rajdhev[1], D. Stalin David[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

The Internet of things makes smart objects the ultimate buildings blocks in the development of cyber-physical smart pervasive frameworks. The IoT revolution is redesigning modern health care with promising technological, economic and social prospects. This paper surveys advances in IoT based health care solutions.In addition,this paper analyses distinct IoT security and privacy features, including security requirements, threat models and attack taxonomies from the health care perspective. Further this paper proposes an intelligent collaborative security model to minimize security risk. The proposed hierarchical approach clusters the documents based on the minimum relevance threshold. The results show that with a sharp increase of documents in the data set. The search time of the proposed method increases exponentially. Furthermore, the proposed method has advantage over the traditional method in the rank privacy and relevance of retrieved documents.IEEE 802.11 standardization for wireless connectivity is gaining popularity day by day because of its low cost solutions and ease of deployment for providing ubiquitous end- user connectivity. In an IEEE 802.11'Wireless Fidelity' (Wi-Fi) network, mobile users can connect to the Internet through wireless access points (APs) that form a backbone network through the wired distribution system. IEEE 802.11 Wi-Fi technology is widely used to provide public wireless connectivity at airports, restaurants and other such public places, known as 'Wi-Fi Hot Spots'. The modern era of 'wireless divide' is gradually witnessing deployments of more advanced wireless technologies, such as IEEE 802.16 or Worldwide Interoperability for Microwave Access (Wi-Max). Wi-Max provides longer coverage area compared to Wi-Fi Technology through advanced modulation and coding schemes for signal transmissions. WiFi-WiMax integration is research topic for next generation wireless Internet architecture and 'Internet of Things'(IoT) designs that attract significant attentions among the researchers. The recent developments in WiFi-WiMAX integration. The current mobile communication system, especially voice communication system is so very well acquainted with the GSM system that it would seem hard to introduce path breaking changes into the way the GSM networks are run. GSM networks use radio frequency (RF) based carriers for mobile communication. The heavy dependency on RF is not only causing severe bandwidth crisis but is also exerting a tremendous pressure on the existing energy generation units. With the number of mobile subscribers increasing exponentially and IoT (Internet of Things) connected devices storming into the market, there is a need for a huge bandwidth that will support all the systems. Visible light communication (VLC) has been proved to be an effective alternative that will prevent the impending crisis. In the past few years, many researchers have come up with VLC based solutions as alternatives to RF in mobile networks. But all of them deal with using VLC in an internet based networking scenario, especially indoor communication. This paper are expected to be useful for researchers,engineers,health professionals and policy makers working in the area of the IoT and health care technologies and also it provides detailed research activities concerning how the IoT can address pediatric and elderly care chronic disease ,private health and fitness management.

Keywords : Local area network, Wide area network, Close circuit television, blood pressure, Just in time.

I. INTRODUCTION

IoT is the network of physical objects with unique identities capable of collecting and exchanging data. Internet of Things has four differential features: only for the thing's information, coded by UID/EPC, stored in RFID electronic tag, and uploaded with non-contact reading of RFID reader.The IoT has the different

architectures based on application support, public & private IP infrastructure, cloud computing, innovation and standards. One of the IoT architecture is RESTful architectures that are becoming one of the most ubiquitous and lightweight integration platforms [8]. Architecture is DIAT and it is an architecture that implements security and privacy simple, scalable architecture for the IoT. It accommodates heterogeneous objects and usage control policies.

The IoT is used to exchange the data from machine to machine, machine to user, things to things and things to user. It is also used to store the devices information. The information is sending from device to IoT server at that time hacking is possible. Hackers are using many techniques to steal passwords at the time of communication.

PrefDB, a preference-aware relational system that transparently and efficiently handles queries with preferences. In its core, PrefDB employs a preference-aware data model and algebra, where preferences are treated as first-class citizens. We define a reference using a condition on the tuples affected, a scoring function that scores these tuples, and a confidence that shows how confident these scores are. In our data model, tuples carry scores with confidences. Our algebra comprises the standard relational operators extended to handle scores and confidences. For example, the join operator will join two tuples and compute a new score-confidence pair by combining the scores and confidences that come with the two tuples. In addition, our algebra

contains a new operator, prefer, that evaluates a preference on a relation, i.e., given as inputs a relation and a preference on this relation, prefer outputs the relation with new scores and confidences. During preference evaluation, both the conditional and the scoring part of a preference are used. The conditional part acts as 'soft' constraint that determines which tuples are scored without disqualifying any tuples from the query result. In this way, PrefDB separates preference evaluation from tuple filtering. This separation is a distinguishing feature of our work with respect to previous works. It allows us to define the algebraic properties of the prefer operator and build generic query optimization and processing strategies that are applicable regardless of the type of reference specified in a query or the expected type of answer. Several approaches to integrating preferences into database queries have been proposed and can be roughly divided into two categories. Plug-in approaches operate on top of the database engine and they typically translate preferences into conventional query constructs. On the other hand, native approaches focus on supporting more efficiently specific queries, such as top-k or skyline queries, by injecting new operators inside the database engine. Unfortunately, both approaches have several limitations. In plug-in methods, the way preferences will be used, for example as additional query constraints or as ranking constructs, the query execution flow as well as the expected type of answer (e.g., top-k or skyline) are all hard-wired in the method, which hinders application development and maintenance. On the other hand, native methods consider preference evaluation and filtering as one operation. Due to this tight coupling, these methods are also tailored to one type of query. Furthermore, they require modifications of the database core, which may not be feasible or practical in real life. Overall, both native and plug-in approaches do not offer a holistic solution to flexible processing of queries with preferences.

II. THE PROPOSED SYSTEM

PrefDB is a prototype system that is based on the preference and extended relational data and query models that we presented earlier. Section 2 provides an overview of its functionality and architecture and also describes the implementation of p-relations and the operators. Query processing in PrefDB Figure 2 depicts the system's architecture. Modules depicted in yellow are provided by the native DBMS, whereas the blue-colored ones are those developed for PrefDB. As shown, PrefDB offers two alternative query options: preferences can be provided along with the input query or the system can enrich a non-preferential query with related preferences. In the first query option, preferences are specified in a declarative way, additionally to the standard SQL query part. In the second case, relevant preferences are provided by the profile manager module, which accesses user preferences stored in the database. Stored preferences can be collected from user ratings or by analyzing past queries or clickthrough data [7]. Since preference collection is orthogonal to query processing, which is the primary goal of PrefDB, in our implementation, we simply store preferences specified by users through a

visual tool we have developed [7] as well as preferences specified in past Query Parser Query + Preferences Query Optimizer Extended Query Plan SQL Execution Engine Database Engine Scoring, aggregate functions Data Operators σ, π, λ, Optimized Query Plan Profile manager Query + Preferences 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.

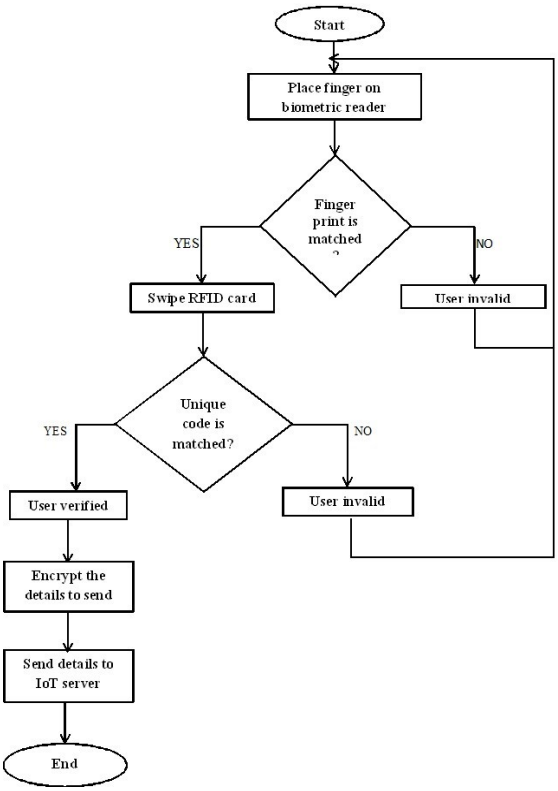


Figure 1. System Architecture

- 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

III. 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.

IV. 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

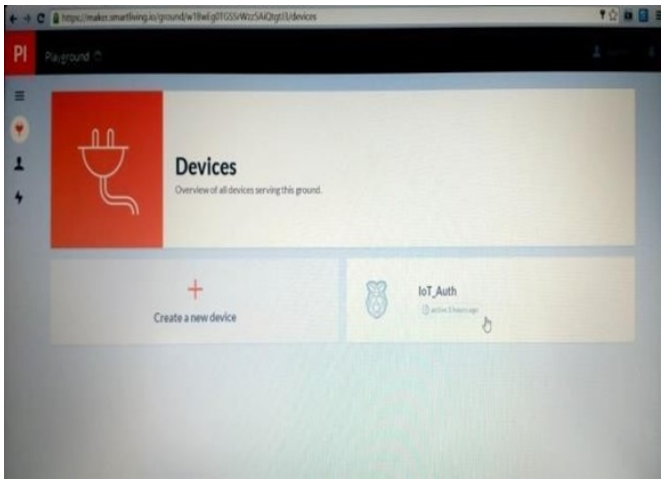
Query Optimization & Execution

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. 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 the score tables is minimized. The execution engine of PrefDB is responsible for processing a preferential query and supports various algorithms.

V.EXPERIMENTAL RESULTS

Sign in to the IoT server.

Click on IoT_Authentication



Attendance details of the employees.

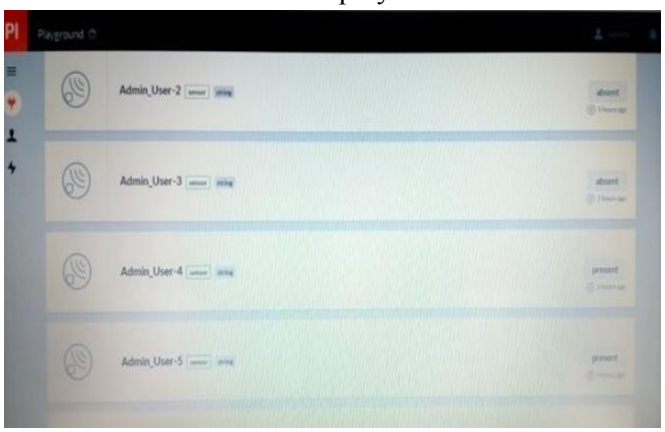


Figure 2. Output of the IoT testbed no finger is placed on the fingerprint module

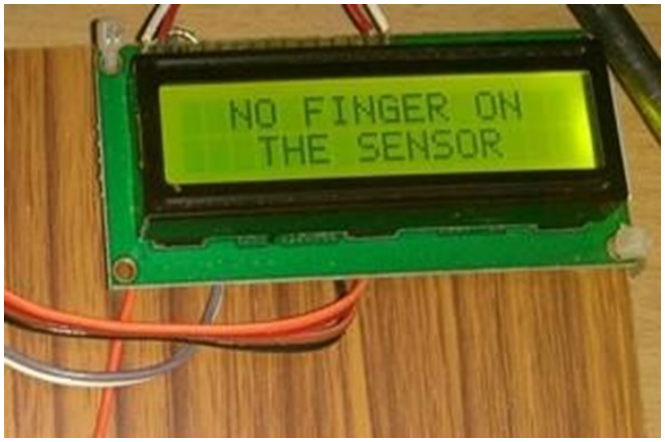
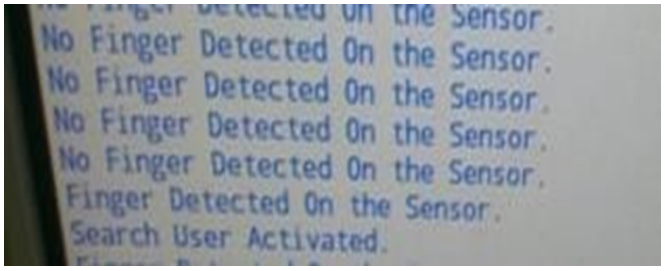


Figure 3. Output of the IoT testbed finger is placed on the fingerprint module



II. CONCLUSION

In this project, surveys diverse aspects of IoT based health care technologies and present various healthcare network architectures and platforms that detailed research activities concerning how the IoT can address pediatric and elderly care, chronic disease supervison, private health and fitness management. This paper presents ehealth and IoT policies and regulations for the benefit of various stakeholders interested in assessing IoT based health care technologies. In sum, the result of this survey are expected to be useful for researchers, engineers, healthprofessionals and policy makers working in the area of the IoT and health care technologies support access to the iot backbone and facilate medical data transmission and reception. The result of this project is expected to be useful for researchers, engineers, healthcare professionals and policy makers working in the area of the IoT and health care technologies.

III. 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.