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

Novel based Intelligent Parking System

A. Vignesh[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

Smart parking system selects desired parking lot, the system will reconnect the driver to the subsystem in related parking lot, the driver can complete the reservation without communicating with the central system. Therefore, the central system no longer needs to maintain the reservation service. Data Collection and Local Presentation: The system collects and stores the data about the performance metrics, including the status of parking space, reservation time, parking location, driver's identity. All data stored by the system is at least stamped with time metadata. Furthermore, the system allows the driver to check the parking information, including the location of parking spaces, the vacancy time of parking spaces and reservation information. In order to protect the security of the system, we separately design a repository of sensing data and a mirror database of reservation. The repository is the sink of the sensing data, and the mirror database is synchronized with the repository and stores the reservation information. In this way, the drivers are only able to check and update the information in the mirror database.

Keywords : Arduino, Wi-Fi (ESP 8266), Load cell, Database System

I. INTRODUCTION

During intervals between allocation decisions made by the center, drivers with no parking assignment have the opportunity to change their cost or walking-distance requirements, possibly to increase the chance to be allocated if the parking system is highly utilized (it is of course possible that no parking space is ever assigned to a driver). The realization of such a "smart parking" system relies on three main requirements. First, the allocation center has to know the status of all parking spots, the location of all vehicles issuing requests and traffic situations. As already mentioned, current sensing technologies make monitoring parking spots implementable. . A "softer" scheme is to use a red/green light system placed at each parking spot, where red indicates that the spot is reserved and only the vehicle assigned to it may switch it back to green (a vehicle parked when the light is red is fined.) In what follows, we will not deal with technical details for meeting these three implementation requirements and concentrate instead on the methodology that enables us to make optimal parking space allocations and reservations.

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. The parking lot consists of a group of parking spaces. The on-street parking can also be considered as a virtual parking lot. The state of a parking lot is the number of occupied spaces versus total spaces. Every parking lot has access to the Internet to communicate with the management system and users, and share parking information with other parking lots. In each parking lot, the reservation authority is deployed for authenticating the individual user's identity and reservation request. In this case, the reservation authority in the parking lot communicates with the specific user individually. Once the reservation order is confirmed, the reservation authority updates reservation information to hold the related space for the user. The sensor system deployed in parking lot is responsible for monitoring the real-time condition of parking lots and delivers the live aggregated sensing information (the number of available spaces or occupancy rate) to the smart parking system. The sensing information is updated on demand. Upon retrieving the parking information, the system updates the state of the parking lot. 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

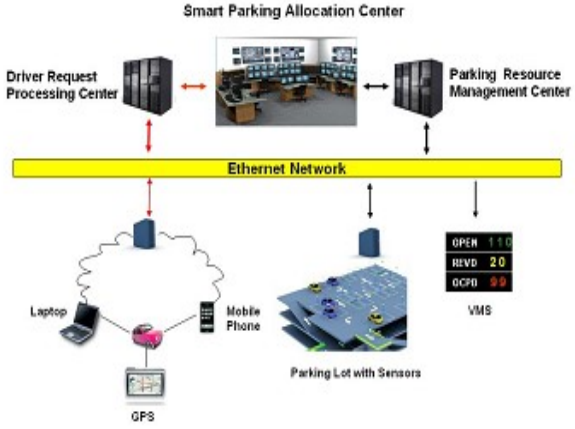


Fig. 1. Smart Parking Framework Overview

- 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. This system explicitly allocates and reserves optimal parking spaces to drivers, as opposed to simply guiding them to a space that may not be available by the time it is reached. The allocation is based on the user's objective function that combines proximity to destination and parking cost, while also ensuring that the overall parking capacity is efficiently utilized. Building on the results in [8], in this paper we refine the allocation algorithm to incorporate features making it suitable for real-world parking problems identified in carrying out a case study based on parking at part of the Boston University campus. Finally, 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.

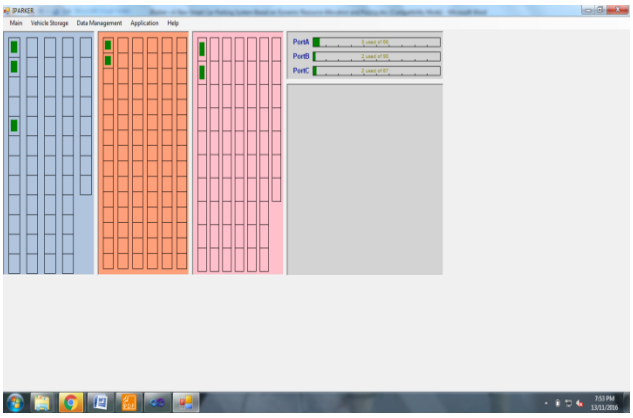
Modules:
Registration & Interest Sum up
Query Formation
Query Optimization & Execution

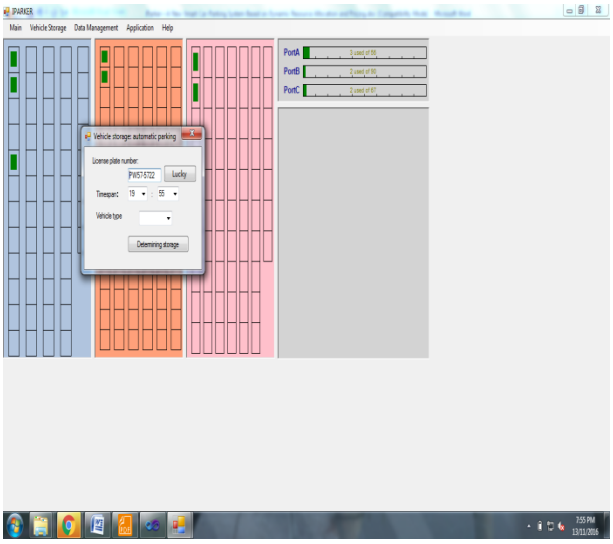
A preferential query combines p-relations, extended relational and prefer operators and returns a set of tuples that satisfy the boolean query conditions along with their score and confidence values that have been calculated after evaluating all prefer operators on the corresponding relations. Intuitively,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

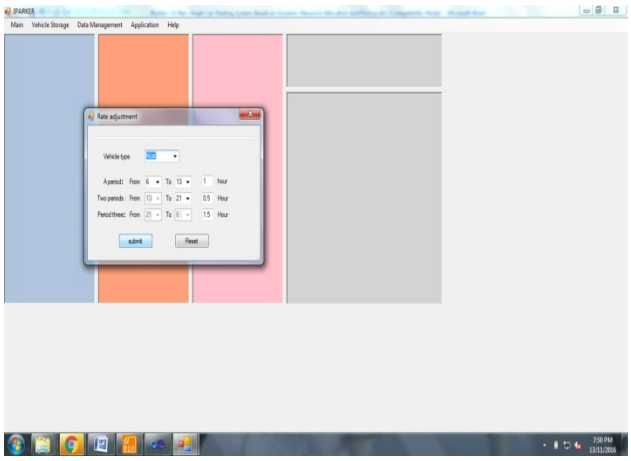
an attribute (either in the conditional or the scoring part) that does not appear in the query and modifies project operators, such that these attributes will be projected as well as proportional to the number of tuples flowing through the operators in the query plan. Assuming a fixed position for the 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.

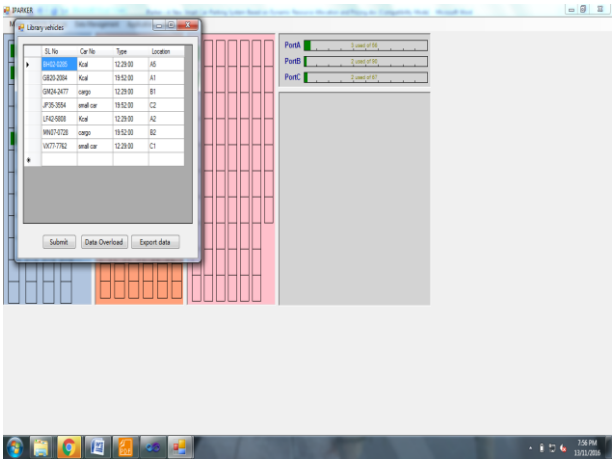
V. EXPERIMENTAL RESULTS

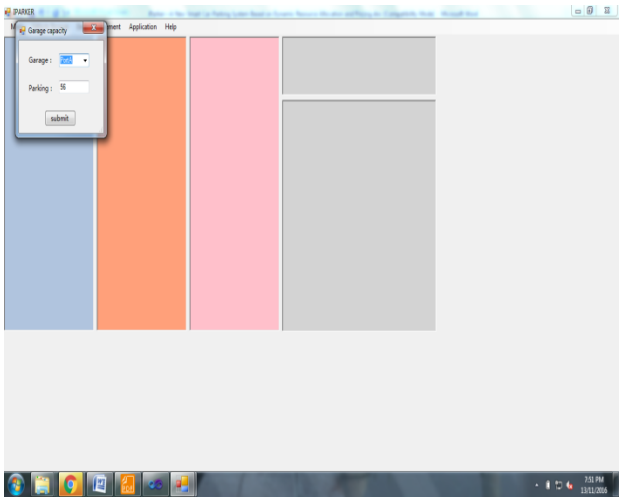
The implementation results can be shown as figure below

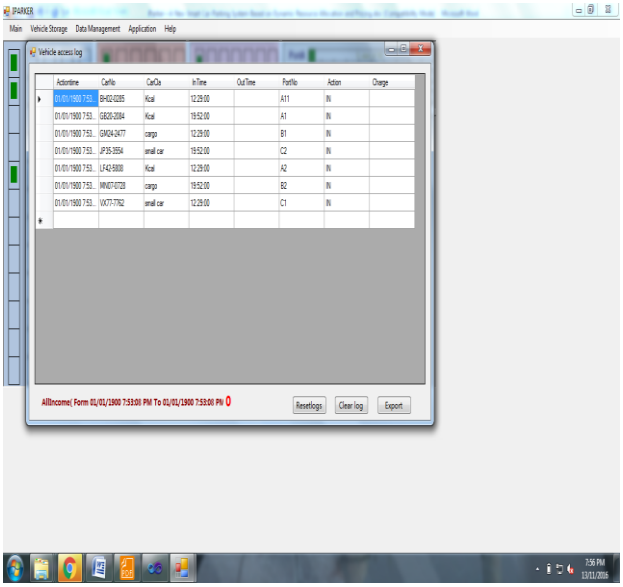












II. CONCLUSION

In this project, In this paper we have proposed iParker, a new smart parkingsystem which is based on MILP model that yields optimal solution for dynamically and statically allocating parking resources to parkers—providing flexible reservation options. The new concepts introduced in this paper are the combination of real-time reservations with share-time reservations, dynamically performing system decisions (reservation time constraints and pricing) according to real-time utilization information, and offering the drivers the choice of choosingmultiple destinations and reservation type.We also have proposed pricing policies for both static and dynamic reservations that maximize the profitfrom parking. Extensive simulation results indicate that theproposed system significantly cuts the total effective cost for all parkers by as much as 28%, maximizes the total utilization by up to 21% and increases the total revenue for parking management up to 16% as compared to the non-guided parking system. Finally we proposed a dynamic pricing scheme and by integrating it to iParker's model, we found by simulations that it balances the utilization across all the parking resources and thus assist in eliminating the overall traffic congestion caused by parking. Currently, the research focuses on a new parking sensing infrastructure and an indoor navigation service for car parking. In the future, we aim to evaluate our system using real-time data and greater number of resources and destinations. In addition, a scalability analysis is to be performed to examine the efficiency of the proposed scalability techniques.

III. REFERENCES

[1]. R. E. Knack, "Pay as you park," Planning, vol. 71, no. 5, pp. 4–8, May 2005.

[2]. National Travel Survey England, Department for Transport, London, U.K., Sep. 2, 2015. Online]. Available: https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/457752/nts2014-01.pdf.

[3]. D. C. Shoup, "Cruising for parking," Transp. Policy, vol. 13, no. 6,pp. 479–486, Nov. 2006.

[4]. A. le Fauconnier and E. Gantelet, "The time looking for a parking space: Strategies, associated nuisances and stakes of parking management in france," in Proc. ETC, Sep. 2006, pp. 1–7.

[5]. IBM Global Parking Survey: Drivers Share Worldwide Parking Woes,IBM, Armonk, NY, USA, Sep. 28, 2011. Online]. Available: https://www-03.ibm.com/press/us/en/pressrelease/35515.wss.

[6]. D. C. Shoup, "The high cost of free parking," J. Plann. Educ. Res., vol. 17,no. 1, pp. 3–20, Fall 1997.

[7]. K. Mouskos, J. Tvantzis, D. Bernstein, and A. Sansil, "Mathematical formulation of a deterministic Parking Reservation System (PRS) with fixed costs," in Proc. 10th MELECON, 2000, vol. 2, pp. 648–651.

[8]. Y. Geng and C. Cassandras, "New smart parking system based on resource allocation and reservations," IEEE Trans. Intell. Transp. Syst., vol. 14, no. 3, pp. 1129–1139, Sep. 2013.

[9]. SFpark, 2015. Accessed on: Feb. 30, 2015. Online]. Available: http://sfpark.org/.

[10]. Y. Ji, W. Guo, P. Blythe, D. Tang, and W. Wang, "Understanding drivers'perspective on parking guidance information," IET Intell. Transp. Syst., vol. 8, no. 4, pp. 398–406, Jun. 2014.

[11]. Y. Asakura and M. Kashiwadani, "Effects of parking availability information on system performance: A simulation model approach," in Proc. IEEE Veh. Navig. Inf. Syst. Conf., 1994, pp. 251–254.

[12]. T. Rajabioun and P. Ioannou, "On-street and off-street parking availability prediction using multivariate spatiotemporal models," IEEE Trans. Intell. Transp. Syst., vol. 16, no. 5, pp. 2913–2924, Oct. 2015.

[13]. K. C. Mouskos, "Technical solutions to overcrowded park and ride facilities," City Univ. New York, New York, NY, USA, Tech. Rep. FHWANJ-2007-011, 2007.
[14]. M. Idris, Y. Leng, E. Tamil, N. Noor, and Z. Razak, "Park system: A review of smart parking system and its technology," Inf. Technol. J., vol. 8, no. 2, pp. 101–113, Mar. 2009.
[15]. G. Revathi and V. Dhulipala, "Smart parking systems and sensors: A survey," in Proc. ICCCA, Feb. 2012, pp. 1–5.
[16]. R. Ranjini and D. Manivannan, "A comparative review on car parking technologies," Int. J. Eng. Technol., vol. 5, no. 2, pp. 1763–1767, Apr./May 2013.
[17]. P. Trusiewicz and J. Legierski, "Parking reservation—Application dedicated for car users based on telecommunications APIs," in Proc. FedCSIS, Sep. 2013, pp. 865–869.
[18]. K. Inaba, M. Shibui, T. Naganawa, M. Ogiwara, and N. Yoshikai, "Intelligent parking reservation service on the internet," in Proc.