Proceedings of the 2010 IEEE
International Conference on Information and Automation
June 20 - 23, Harbin, China

A Sample-Guided Approach to Incremental Structured Web Database Crawling

Wei Liu, Jianguo Xiao, Jianwu Yang

Institute of Computer Science & Technology
Key Laboratory of Computational Linguistics (Peking University), MOE
Peking University
Beijng China, 100871

{liuwei, xjg, yangjianwu}@icst.pku.edu.cn

***Abstract*—Web database crawling is a promising solution for Deep Web data integration. To the best of our knowledge, the existing approaches only focused on how to crawl all records in a web database. Due to the high dynamic of most web databases, it is not practical to harvest a small proportion of new records by crawling the whole database. This paper studies the problem of incremental web database crawling, which targets at crawling the new records from a web database efficiently. In the proposed approach, a new graph model, query related graph, is proposed to transform a incremental crawling task into a graph traversal process. Based on this graph model, appropriate queries are generated for crawling which are guided by the samples of the web database. Extensive experimental evaluations over real Web databases validate the effectiveness of our techniques and provide insights for future efforts in this direction.**

Index Terms - Web database, Deep Web data integration, Web database crawling

I. INTRODUCTION

The Deep Web refers to the data residing in web databases, and most of its content is in form of structured data records[1]. The Deep Web is believed to be the largest source of structured data on the Web and hence Deep Web data integration has been a long standing challenge in the field of Web data management. A promising solution for Deep Web data integration is web database crawling[2]. Crawling-based solution targets at gathering structured records from web databases to make users search and mine the Deep Web in a centralized manner. The rapid development of computer hardware and Internet makes this solution more practical than before.

To the best of our knowledge, previous efforts[3][4][5][6] only focus on crawling the whole web database with the goal of maximizing the coverage of the web database. We call this approach "exhaustive crawling". As it is widely known, most web databases are highly dynamic, e.g. new records are always being inserted constantly. To assure the local database is consistent to the integrated web databases, the maintenance operation has to be performed. However, it is not affordable to always apply the exhaustive-crawling approach to harvest a small quantity of new records(compared to the whole web database), which can result in the heavy burdens for both web databases and the network. In this paper, we study a crucial but largely unresolved problem in the crawling-based solution: how to obtain the new records without crawling the whole web database?

To this end, we propose a sample-guided incremental-crawling approach. The basic idea of this approach is described as follows. First, a small number of random samples are harvested from the web database. Then, by analyzing the deviation between the samples and the history version of the web database, an appropriate record is selected to generate the promising query for crawling new records. In this approach, we propose query-related graph model, and hence, any given web database can be represented as an undirected graph based on the model. The incremental crawling task is thus transformed into a graph traversal process in which the crawler starts with the graph of the samples of the web database and at each step a vertex v is selected and an appropriate query is generated using the selected vertex for crawling. Since the only general way of accessing a web database is through its query interface, automatic query generation is the key of our approach. Our goal is to maximize the coverage of the new records and minimize the coverage of the old ones of at the same time.

As the initial effort to address the incremental web database crawling problem, the contribution of the paper is summarized as follows. First, we identify this novel problem of incremental web database crawling. Contrary to the previous exhaustive-crawling works, we demonstrate that a central issue of efficient web database crawling lies in the consistency between the local database and the integrated web databases. Second, we provide a theoretical framework that formally models query-based web database crawling as graph traversal. Different to the attribute-level graph models proposed by previous works(e.g. [3]), our graph model is on record level, which can characterize whether any two records are query related in a straightforward way. Third, based on the graph model, we propose simple and smart methods for the key problems in the incremental-crawling approach, which aims at generating promising queries to harvest the new records as many as possible.

The rest of this paper is organized as follows: Section 2 presents the preliminaries. Section 3 introduces the query-related graph model. The query selection method based on the query-related graph model is proposed in Section 4. We discuss our experimental findings in Section 5. Section 6 reviews some related work. Section 7 concludes this paper.

978-1-4244-5704-5/10/$26.00 ©2010 IEEE

II. PRELIMINARIES

A. Samples of WDB

The samples of WDB refers to a set of records which are selected randomly from WDB. Since the only general way of accessing a web database is through its query interface, traditional random sampling techniques cannot be easily applied as we do not have full access to *WDB*. Fortunately, [17] proposed a random walk approach to sampling web database. In this approach, probabilistic rejection based techniques are used to improve the sample quality, and the size of samples can be tuned by setting the parameters. In this paper, we assume the samples of have been already obtained. In fact, the samples are impossible to be truly unbiased whatever approaches are applied because the queries for sampling are biased. In the experiments, we will give a discussion for the impact on performance which is brought by biased samples.

B. History Version of WDB

As we have mentioned, the Deep Web is highly dynamic. We use WDB_h to denote the history version of *WDB*, i.e. the records of *WDB* which were crawled previously and stored in the local database. In this paper, we only consider the scenario of record insertion which is common in many domains, such as research domain and job domain. We use $WDB\text{-}WDB_h$ to denote the new records being inserted during the period. The history version of *WDB* can be easily obtained from the local database by issuing SQL "select * from *WDB*" to the local database. In this way, the history version of *WDB* can be regarded as a virtual view of the local database.

C. Discretizing the Value Ranges of Attributes

In this paper, we assume that the value range of any attribute in the query interface is a set of discrete values. In fact, there are three types of input choices provided for users to fill a query interface: keyword, selecting and range. "selecting" attribute has already meet this requirement. For "keyword" attribute, its values are all possible none-stop keywords appearing in records on this attribute. As a result, more than one values may appear in the attribute of a record. For example, the title of a research paper usually contains several values(keywords). For "rang" attribute, its value range is divided into a set of small ranges. In this way, all attributes can be processed without difference.

D. Performance of Incremental Crawling

The goal of incremental web database crawling is to crawl as many new records as possible while minimizing the communication costs. As a result, two factors have to be considered: coverage rate and crawling cost. The former is used to measure the effectiveness, while the latter is used to measure the efficiency.

Coverage rate

In the exhaustive-crawling works, coverage rate refers to the ratio between the number of crawled unique records and all the records of the web database, and it is denoted as *TCR* (**T**otal **C**overage **R**ate) in this paper. For the incremental web database crawling, the ratio between the number of crawled unique new records and the number of all new records is also considered, and it is denoted as *ICR*(**I**ncremental **C**overage **R**ate). Formally, the definitions of them are given below:

$$TCR=\frac{|\text{Crawled New Records}|+|WDB_h|}{|WDB|} \quad (1)$$

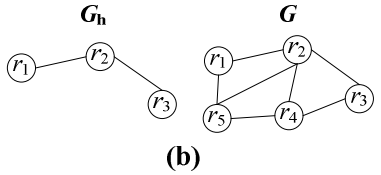
WDB_h

	A	B
r_1	a1	b1, b2
r_2	a2	b2, b3
r_3	a3	b3

WDB

	A	B
r_1	a1	b1, b2
r_2	a2	b2, b3
r_3	a3	b3
r_4	a3	b3, b4
r_5	a2	b2, b5

(a)



(b)

Figure 1. An example to illustrate the query related graph model

$$ICR=\frac{|\text{Crawled New Records}|}{|WDB)|-|WDB_h|} \quad (2)$$

Crawling cost

The crawling cost in this paper refers to the proportion of the new records in all the of returned records(including the duplicated records crawled with different queries) during the crawling process, and it is denoted as *PNR*(**P**roportion of **N**ew **R**ecords). The formal definition is given below:

$$PNR=\frac{|\text{Crawled New Records}|}{|\text{All Crawled Records}|} \quad (3)$$

Obviously, given *CCR* and *ICR*, larger *PNR* is, more efficient the crawler is. For instance, if 100 new records have been crawled and 1000 records were returned by *WDB* after the incremental process, then *PNR*=0.1. To this end, we must try to elaborately select the queries which can harvest more new records and maximize the ratio between the crawled new records and the number of all crawled records.

III. QUERY-RELATED GRAPH MODEL

A *query-related graph* (QRG), *G(V,E),* for *WDB* is an undirected graph that can be constructed as follows: for each record r_i, there exists a unique vertex $v_i \in V$. An undirected edge $(v_i,v_j) \in E$ i.f.f. r_i and r_j satisfy at least one query q that can be represented in the query interface, i.e. both r_i and r_j are in the query result of q.

According to the definition of this model, the graph complexity of a given *WDB* is determined by two factors: the number of records and the query capability of the query interface. Here the query capability refers to the number of attributes contained in the query interface and the value ranges of attributes. In other words, more attributes and more large size of the value ranges of the attributes will result in more edges in the graph. An interesting property of QRG is: if a vertex v is selected for query generation, the corresponding

vertexes of the returned records must be some of the neighbors of v. Intuitively, if a vertex has many neighbors, more records are possible to be returned. We will use this character for vertex selection by making an approximate estimation for the number of the returned records.

By characterizing structured web databases using QRG, a incremental crawling task is transformed into a graph traversal process in which the crawler starts with the graph of the history version of *WDB* and at each step a vertex v is selected for crawling(query formulation) and the new records which are the neighbors of v will be harvested and stored for future crawling.

Fig. 1 shows an example to illustrate the proposed graph model. Fig. 1(a) shows *WDB* and WDB_h, each record consist the values over two attributes A and B. Fig. 1(b) is the query-related graphs of *WDB* and WDB_h respectively. And r_4 and r_5 are the new records in *WDB*. There are different choices to crawl r_4 and r_5. For example, one choice is selecting r_1 and then using "b2" on B to crawl r_5, and selecting r_3 and using "a3" on B to crawl r_4, another choice is selecting r_2 and using "b2" and "b3" on B to crawl r_4 and r_5. Obviously, different choices result in different coverage rate and crawling cost. In the next section, we will present our approach to crawl the new records based on QRG by making good choices.

IV. Approach of Incremental Web Database Crawling

A. Approach Overview

Incremental Web Database Crawling Algorithm
Input: S, WDB_h //The samples of *WDB* and the history version of *WDB*
Output: *NRS* // the set to store the crawled new records
Begin
1 Produce the query-related graphs G_S and G_h for S and WDB_h respectively;
2 While the terminate criteria is not met
3 r=RecordSelection(G_S, G_h);
4 q=QueryGeneration(r);
5 *RS*=RecordExtraction(q);
6 *NRS*.Add(*RS*);
7 Return *NRS*;
End

Figure 2. The overview of the proposed approach

We first overview the proposed approach. The basic idea of our approach is to crawl the new records from *WDB* by analyzing the deviation between the samples and the history version of the web database. We assume the unbiased samples of *WDB* have been obtained and stored in the local database. We use S to denote the samples. Formally, our problem can be depicted as follows: given S and WDB_h, how to get the new records$\in$ *WDB*- WDB_h?

Fig.2 shows the overview of our approach, which is a loop process. The functions of the three components in each round are listed below.

RecordSelection: select a record from $S \cup WDB_h$.

QueryGeneration: generate a query using the selected record.

RecordExtraction: extract the records from the result pages.

Record selection and query generation are the key components in our approach. Record extraction belongs to the research field of web data extraction, which has been widely studied, so no more discussion is given for it in this paper. The stop criteria of the crawling process is no new records were crawled within 10 queries. In the rest part of this section, we will introduce the underlying techniques for record selection and query generation.

B. Record Selection

Record Selection Algorithm
Input: G_s, G_h
Output: v // the selected vertex in $G_S \cup G_h$
Begin
1 For each vertex v in $G_S \cup G_h$ do
2 Compute $Sel(v)$;
3 Sel_Sum=$\sum Sel(v_i)$ $v_i \in G_S \cup G_h$
4 For each vertex v in $G_S \cup G_h$ do
5 Prob(v)=$\frac{Sel(v)}{\text{ACT\_Sum}}$; //the probability of selecting v
6 Randomly select a vertex v from $G_S \cup G_h$ according to Prob(v);
7 Return v;
End

Figure 3. Record Selection Algorithm

When the query-related graphs G_S and G_h have been produced, the record selection problem is how to select a vertex from $G_S \cup G_h$ whose neighbors are new records as many as possible. According to the definition of *PNR*, the selected vertex at each round must make the proportion of the new vertexes among its neighbors in G as much as possible. In other words, the distance of the selected vertex and the new vertexes(i.e. the vertex in G-G_h) must be as near as possible and the distance of the selected vertex and the old vertexes(i.e. the vertex in G_h) must be as far as possible. However, G is not known yet, and we use G_S as the representative of G under the consumption that S is unbiased. Based on such consideration, the vertex selection criteria is represented with the following formula.

$$Sel(v)=\frac{\text{ShortestPath}(v, G_h)}{\text{ShortestPath}(v, G_S)} \quad (4)$$

where ShortestPath(v, G_h) refers to the average shortest path between v and the vertexes in G_h, and ShortestPath(v, G_h) refers to the average shortest path between v and the vertexes in G_S. As a result, given a vertex v, larger $Sel(v)$ is, more possible v is to connect to new vertexes. With Equ. (4), the algorithm of record selection is given in Fig. 3.

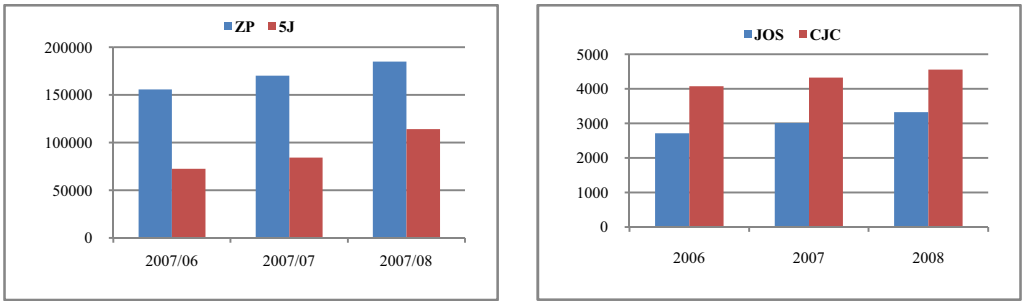
C. Query Generation

After a vertex(record) has been selected, we will generate an appropriate query using the vertex. Each query can be modeled as a set of equality predicates, such as the query "position=Software Engineer and company=IBM" for searching jobs. Same to the existing works on web database crawling, we also focus on the simplest selection queries with only one equality predicate. Given a record, multiple queries can be generated. For example, three queries "A=a1", "B=b1", and "B=b2" can be generated using r_1 in Fig. 1.

Considering the crawling cost measure *ICR*, the goal of query generation is to select the most effective one from all possible queries. We define the effectiveness of a query below:

TABLE I
The dataset used in our experiment

Web database	Home page	Queriable Attributes	Start time	End time
Zhaopin.com(ZP)	www.zhaopin.com	Position, Company, Location	2007/06/01	2007/08/31
51Job.com(5J)	www.51job.com			
Journal of software(JOS)	www.jos.org.cn	Title, Author, Key Words	2006/01/01	2008/12/31
Chinese Journal of Computers(CJC)	cjc.ict.ac.cn			



(a) (b)

Fig. 4 The versions of the web databases: (a) Zhaopin and Chinaren from 2007/06 to 2007/08; (b) JOS and CJC from 2006 to 2008.

$$eff(q)=\frac{\text{count}(WDB,q)-\text{count}(WDB_h,q)}{\text{count}(WDB_h,q)} \quad (5)$$

where count(*WDB*, *q*) refers to the number of returned records when issuing *q* to *WDB*. count(*WDB*, *q*) can be got by extracting the hit number from the result page, such as "3,850" in "1 - 10 of **3,850**, (0.093) seconds". According to our statistics to a large number of web databases[16], more than 98.4% web databases present the hit numbers in their result pages to indicate the number of returned records for each query. And a method has been proposed by [15] to extract the hit numbers. count(WDB_h, *q*) can be got by issuing "select count(*) from WDB_h where *q*" to the local database. As a result, count(*WDB*, *q*)-count(WDB_h, *q*) is the number of new records we will obtain.

Based on Eq. (5), a simple method for query generation is described as follows. First, generate all one-equality-predicate queries using the selected record. Second, compute the effectiveness for each query. At last, the query with the max effectiveness is used for crawling.

V. Experimental Evaluation

In this section, we first present the dataset used in our experiment, then a set of experiments are conducted to evaluate the performance of the crawler implemented based on the proposed approach. Both the crawler and the local database are hosted by a 2.66G Windows server with 3.25G RAM running on MySQL 5 RDBMS.

A. Dataset

To evaluate the crawler precisely and objectively, our dataset is the history data of four real web databases from job and research domains which is provided by JobTong(www.jobtong.cn) and C-DBLP(www.cdblp.cn). The four web databases are Zhaopin.com(ZP), 51Job.com(5J), Journal of software(JOS), and Chinese Journal of Computers(CJC). The details of the dataset are shown in Table 1. In Table I, the second column shows the attributes that can be queried in the query interfaces. The third and the fourth columns give all the history versions in our experiment being produced during this period. Fig. 4 shows the sizes of the history versions. x-axis refers to the number of records, and y-axis refers to the time line. For job domain, there are 3 versions for each web database(time interval: one month); for research domain, there are also 3 versions for each web database(time interval: one year). The average growth speed on job domain is about 9.5% per month, and the average growth speed on job domain is about 12.1% per year.

B. Experiment on coverage rate and crawling cost

TABLE II.
EXPERIMENTAL RESULTS ON COVERAGE RATE AND CRAWLING COST

WDB	Measure	07	08
ZP	*CCR*	97.9%	99.1%
	ICR	86.3%	91.5%
	PNR	19.9%	26.2%
5J	*CCR*	97.2%	98.6%
	ICR	80.8%	91.3%
	PNR	28.5%	35.1%

(a) Job domain

WDB	Measure	2007	2008
JOS	*CCR*	96.2%	97.5%
	ICR	64.0%	73.4%
	PNR	18.4%	20.4%
CJC	*CCR*	98.6%	98.7%
	ICR	80.7%	81.9%
	PNR	23.5%	24.6%

(b) Research domain

We evaluate the performance of the crawler on both the coverage rate(*CCR* and *ICR*) and the crawling cost(*PNR*) over the four web databases. A family of experiments are conducted, and Table 2 shows the experimental results. We use the last version as the history version to crawl the next version. For example, the head of the 3th column of Table 2(a) means we use 06 version to crawl the new records in version 07. The

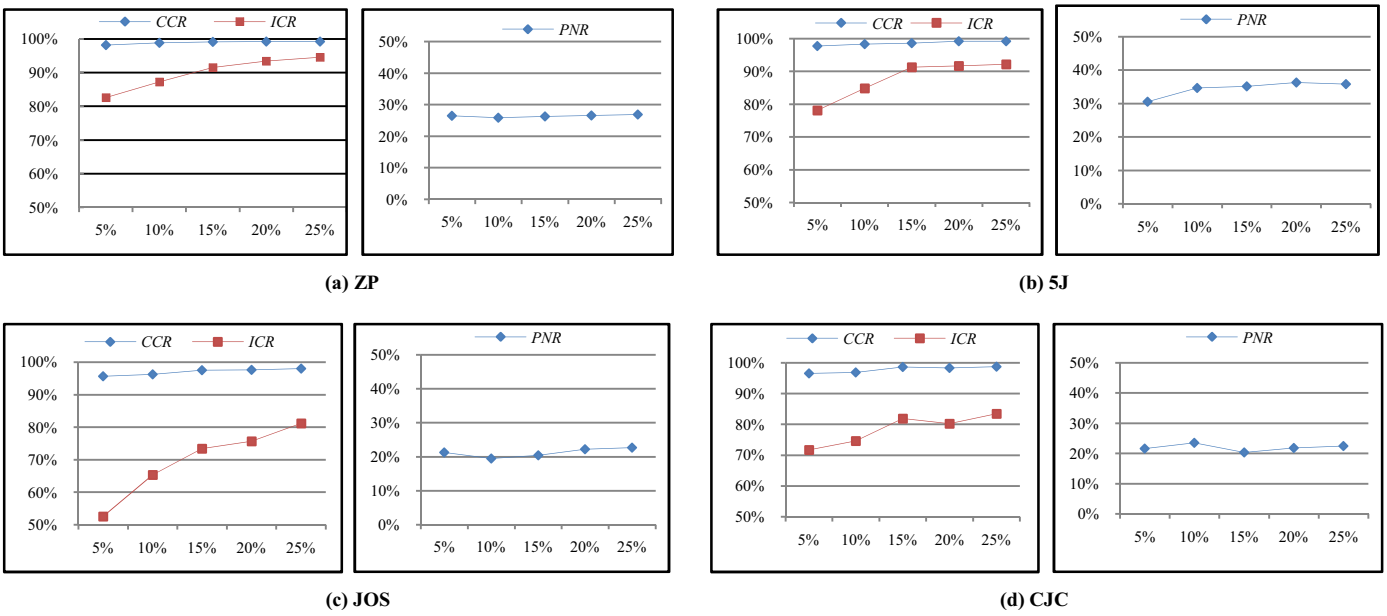


Figure 4. Experiment on different size of samples

ratio of the samples in each web database is set as 15%. The samples are selected randomly As it can be seen from Table 2, the crawler based on our approach can achieve very high performance on both coverage rate and crawling cost. For *CCR*, all ones are larger than 96% and some even close to 100%. This means almost all the records in current version of *WDB* can be got by us. For *ICR*, the ones(>80%) on job domain are better the ones(>64%) on research domain. We think the reason is that the sizes of job web databases are much larger than thoseof research web databases, which makes more edges exist among the records in their query related graphs. For *PNR*, all ones are in a stable range(job domain: 19.9%-35.1%; research domain: 18.4%-24.6%). This indicates that: (1)the crawler is very robust, which means its performances will not fluctuate significantly over different web databases; (2) the proportion of the new records in all the crawled records.

Compared with the approaches on exhaustive crawling, our incremental crawling approach is significantly superior to them on the measures *CCR* and *PNR*. To the best of our knowledge, [3] is the state-of-art work on exhaustive crawling. The best performance on *CCR* reported in its experimental result is about 90%, which is far less than our average *CCR*(98%). According to the crawling cost reported in its experimental result, the returned records is 1.2-1.5 times of the crawled unique records. If the growth speed of *WDB* is 10%, its *ICR* is (1.1-1)/(1.1*1.2)<10%.

C. Experiment on different size of samples

In the last experiment, the ratio of the samples in each web database is set as 15%. To evaluate the influence of this ratio on the performance, we set the ratio as different values and carry out the incremental crawling experiment again. For job domain, the crawling target is 08 version; for research domain, the crawling target is 2008 version. Fig. 5 shows the experimental results on different ratios. There are two conclusions can be made. First, the size of samples has the distinct influence to *CCR*, *ICR*. In general, when the ratio is larger than 15%, *CCR* and *ICR* tend to be stable. So the ration can be set as 15% in practice because obtaining a large size of samples is also a heavy burden. Second, there is no evidence that the ratio has an influence on *PNR*. The reason is more versions will make a better choice for record selection, which can improve the proportion of new records in the returned records of each query.

D. Experiment on biased samples

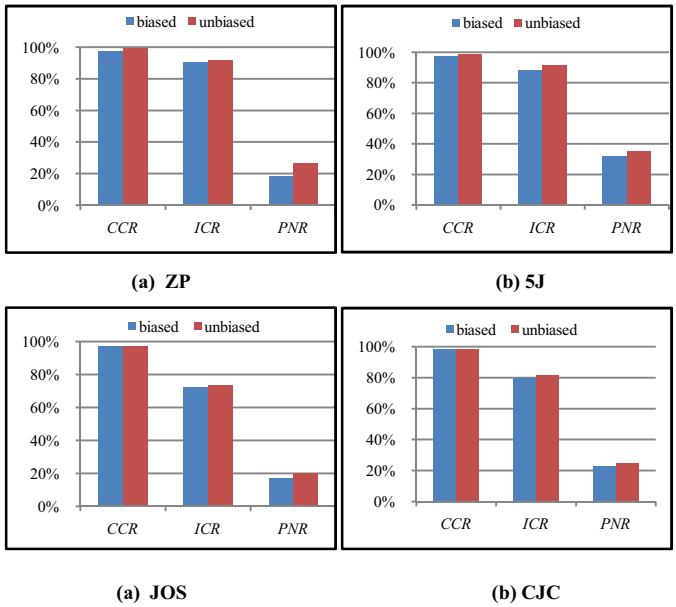


Figure 5. Experimental results on biased samples

In practice, it is impossible to obtain real unbiased samples, so it is necessary to evaluate our approach on approximate unbiased samples. We adopt the naïve method proposed in [17] to obtain the approximate unbiased samples. The ratio is also set as 15%. Form the experimental results shown in Fig. 6, we

can find that the overall performance over unbiased samples is slightly better than that over approximate unbiased samples. This indicates that our approach on approximate unbiased samples can also achieve a high performance.

VI. Related Work

[4] is the first work on web database crawling, which is presented to automatically extract and analyze the interface elements and submits queries through these query interfaces. [7] studies the construction of keyword queries to obtain documents from large web text collections. [8] proposes similar keyword query selection techniques for downloading the textual content from web databases. [5] reduces the problem of selecting an optimal set of queries from a sample of the data source into the well-known set-covering problem. [6] models web database crawling as a set covering problem and developed a new set covering algorithm to address this problem. All the works above focus on how to crawl the whole web database, which is not practical for highly dynamic web databases. Different with them, we study the incremental crawling problem. Though being a follow-up work to previous works, it is indispensable in the real deep web data integration systems. Web database sampling is similar to web database crawling: it also harvests the records from a web database by issuing queries to the query interface. Different to web database crawling, web database sampling targets at obtaining a small proportion of records(not all) which are distributed uniformly in the web database. [17] is the first work on web database sampling. It proposed a random walk approach to sampling web database. In this approach, probabilistic rejection based techniques are used to improve the sample quality, and the size of samples can be tuned by setting the parameters.

Another important related area is web data extraction, which is also a key component in the incremental crawler. Lots of solutions have been proposed for this issue. [9] and [10] study the problem of fully automatic data extraction by exploring the repeated patterns from multiple template generated result pages. [11] utilizes the information on the ‘‘detailed’’ record pages pointed by the current result page to identify and extract data records. Note that Web data extraction is orthogonal to the query selection problem investigated in this paper. Recently, the efforts on this field pay more attention to vision-based solutions which are independent of web page programming languages, such as [18] and [19]. There has been an active research interest in understanding the semantics of the query interfaces of the structured Web databases. [12] introduces WISE-integrator that employs comprehensive meta-data, such as element labels and default value of the elements to automatically identify matching attributes. [13] proposes an instance-based schema matching scheme that uses domain specific query probes to discover the attribute mappings. [14] uses statistical models to find the hidden domain-specific schema by analyzing co-appearance of attribute names.

VII. Conclusion and Future Work

The high dynamic of web databases makes the exhaustive crawling approach impractical to maintain the data consistency between the local database of the Deep Web data integration system and the integrated web databases. In this paper, we studied the incremental web database crawling problem, and proposed an efficient and effective method to address this problem. The extensive experiment on four real web databases shows our approach can significantly reduce the crawling cost without the loss of coverage rate.

As the future work, we will improve the crawling cost by combining the total number of issued queries into *PNR*, which can measure the crawling cost more objectively.

VIII. ACKNOWLEDGMENT

We would like to thank the anonymous reviewers for useful comments. This work was supported in part by the China Postdoctoral Science Foundation funded project under grant 20080440256 and 200902014, NSFC (60875033), National High-tech R&D Program (2008AA01Z421) and National Development and Reform Commission High-tech Program of China (2008-2441). Any opinions, findings, conclusions, and/or recommendations in this material, either expressed or implied, are those of the authors and do not necessarily reflect the views of the sponsors listed above.

References

[1] B. He, M. Patel, Z. Zhang, and K. C.-C. Chang. Accessing the Deep Web: A survey. Communications of the ACM, 50(5) 2007.

[2] J. Madhavan, L. Afanasiev, L. Antova, A. Y. Halevy: Harnessing the Deep Web: Present and Future. In CIDR 2009

[3] P. Wu, J.-R. Wen, H. Liu, W.-Y. Ma: Query Selection Techniques for Efficient Crawling of Structured Web Sources. In ICDE 2006

[4] S. Raghavan, H. Garcia-Molina: Crawling the Hidden Web. In VLDB 2001: 129-138

[5] J. Lu, Y. Wang, J. Liang, J. Chen, J. Liu: An Approach to Deep Web Crawling by Sampling. In Web Intelligence 2008

[6] Y. Wang, J. Lu, J. Chen: Crawling Deep Web Using a New Set Covering Algorithm. In ADMA 2009

[7] L. Barbosa, J. Freire: Siphoning Hidden-Web Data through Keyword-Based Interfaces. In SBBD 2004

[8] A. Ntoulas, P.Zerfos and J. Cho. Downloading textual hidden Web content through keyword queries. In JCDL, 2005.

[9] H. Zhao, W. Meng, Z. Wu, V. Raghavan, C. T. Yu: Fully automatic wrapper generation for search engines. In WWW 2005

[10] Y. Zhai, B. Liu: Web data extraction based on partial tree alignment. In WWW 2005

[11] K. Lerman, L.Getoor, S. Minton and C. Knoblock. Using the structure of Web sites for automatic segmentation of tables. In SIGMOD, 2004.

[12] H. He, W. Meng, C, Yu and Z. Wu. WISE-Integrator: an automatic integrator of Web search interfaces for E-commerce. In VLDB, 2003.

[13] J. Wang, J. Wen, F. H. Lochovsky, and Wei-ying Ma. Instancebased Schema Matching for Web Databases by Domain-specific Query Probing. In VLDB, 2004.

[14] B. He, and K. C. Chang. Statistical Schema Matching across Web Query Interfaces. In SIGMOD, 2003.

[15] Y. Ling, X. Meng, W. Meng: Automated Extraction of Hit Numbers from Search Result Pages. In WAIM 2006

[16] http://idke.ruc.edu.cn/news/2008/dataset.htm

[17] A. Dasgupta, G. Das, H. Mannila: A random walk approach to sampling hidden databases. In SIGMOD 2007