ISSN (ONLINE): 2395-695X
ISSN (PRINT): 2395-695X

International Journal of Advanced Research in Basic Engineering Sciences and Technology (IJARBEST)
Volume 2, Special Issue 19, October 2016

Enhanced Aggregate Estimation in Hidden Databases Using Back Tracking and Divide and Conquire Algorithm

A.Pavithra,
Research Scholar,
Department of Computer Science,
Bharathidasan College of Arts and Science,
Erode.
apavithramca@gmail.com

Ms. S. Sangeetha M.C.A., M.phil,
Assistant Professor,
Department of Computer Science,
Bharathidasan College of Arts and Science,
Erode.
sangee.siva2011@gmail.com

***Abstract*— In this paper describe the problem of estimating the size of a hidden database through its web interface. In this propose system novel techniques which use a small number of queries to produce unbiased estimates with small variance. These techniques can also be used for approximate query processing over hidden databases. present theoretical analysis and extensive experiments to illustrate the effectiveness of proposed approach. Hidden databases are widely prevalent on the web. They feature restrictive form-like interfaces which allow users to form a search query by specifying the desired values for one or a few attributes, and the system returns a small number of tuples satisfying the user-specified selection conditions. In this paper initiate a study of estimating, without bias, the size and other aggregates over a hidden database. For size estimation, our main result is HD-UNBIASEDSIZE, an unbiased estimator with provably bounded variance. For estimating other aggregates, to extend HD-UNBIASED-SIZE to HD-UNBIASED-AGG which produces unbiased estimations for aggregate queries. In addition proposed methodology also concentrates in finding the most favorable and fast stirring items that interested and most wanted by the users. Through the favorable items and customers viewed items, the study tracks the frequent and fast selling items over through the social networking applications and shopping sites in a particular season or time period. To estimate the size of a hidden database, on intuitive idea is to perform tuple sampling.**

Keywords*—Hidden Database, Web Interface, Aggregate Estimation, fast Moving Analysis, Catch base Left Deep tree

I. INTRODUCTION

Data mining involves the use of sophisticated data analysis tools to discover previously unknown, valid patterns and relationships in large data set. These tools can include statistical models, mathematical algorithm and machine learning methods. Consequently, data mining consists of more than collection and managing data, it also includes analysis and prediction. Classification technique is capable of processing a wider variety of data than regression and is growing in popularity.

Web mining is the use of data mining techniques to automatically discover and extract information from Web documents and services. There are three general classes of information that can be discovered by web mining:

- Web activity, from server logs and Web browser activity tracking.
- Web graph, from links between pages, people and other data.
- Web content, for the data found on Web pages and inside of documents.

Web Mining versus Data Mining

When comparing web mining with traditional data mining, there are three main differences to consider:

- **Scale** – In traditional data mining, processing 1 million records from a database would be large job. In web mining, even 10 million pages wouldn't be a big number.
- **Access** – When doing data mining of corporate information, the data is private and often requires access rights to read. For web mining, the data is public and rarely requires access rights.
- **Structure** – A traditional data mining task gets information from a database, which provides some level of explicit structure. A typical web mining task is processing unstructured or semi-structured data from web pages. Even when the underlying information for web pages comes from a database, this often is obscured by HTML markup.

Hidden databases are data repositories hidden behind the only accessible through a restrictive web search interface. Input capabilities provided by such a web interface range from a simple keyword-search textbox to a complex combination of textboxes, dropdown controls, checkboxes, etc. Once a user specifies a search query of interest through the input interface, the hidden database selects and returns a limited number of tuples satisfying the user-specified search conditions.

The static webpages (connected by hyperlinks), the contents of a hidden database cannot be easily crawled by traditional web search engines. In fact, the restrictive web interface prevents users from performing complete queries as they would with the SQL language. For example, there are hardly any web interfaces providing aggregate queries such as COUNT and SUM functions. The lower query capability of a hidden database surely reduces its usability to some extent.

The aggregate estimation consists of two components: bias and variance. The proposed methodology weighted sampling is used to minimize variance. Dynamically adjust the probability of sampling a query based on the query answers we receive so far, in order to align the sampling process to both the data distribution and the aggregate to be estimated, and thereby reduce the variance of our aggregate estimations. The main objective of the paper are,

- To produce unbiased aggregate estimations over the hidden databases with checkbox interfaces, develop the data structure of left-deep-tree and define the concept of designated query to form an injective mapping from tuples to queries supported by the web interface.
- To produce unbiased aggregate estimations over the hidden databases with checkbox interfaces, develop the data structure of left-deep-tree and define the concept of designated query to form an injective mapping from tuples to queries supported by the web interface.
- To reduce the variance of aggregate estimations, develop the ideas of weighted sampling and special tuple-crawling.

II. RELATED WORK

Cheng Sheng et al [1] describe a hidden database refers to a dataset that an organization makes accessible on the web by allowing users to issue queries through a search interface. In other words, data acquisition from such a source is not by following static hyper-links.

- Instead, data are obtained by querying the interface, and reading the result page dynamically generated. This, with other facts such as the interface may answer a query only partially, has prevented hidden databases from being crawled effectively by existing search engines.
- To extract all the tuples from a hidden database and algorithms are provably efficient, namely, they accomplish the task by performing only a small number of queries, even in the worst case.

They also establish theoretical results indicating that these algorithms are asymptotically optimal i.e., it is impossible to improve their efficiency by more than a constant factor. The derivation of our upper and lower bound results reveals significant insight into the characteristics of the underlying problem. Extensive experiments confirm the proposed techniques work very well on all the real datasets examined.

Sriram Raghavan et al [2] describe a crawlers retrieve content only from the publicly indexable Web, i.e., the set of Web pages reachable purely by following hypertext links, ignoring search forms and pages that require authorization or prior registration. In particular, they ignore the tremendous amount of high quality content "hidden" behind search forms, in large searchable electronic databases. They address the problem of designing a crawler capable of extracting content from this hidden Web. They introduced a generic operational model of a hidden Web crawler and described how this model is realized in Hidden Web Exposer, a prototype crawler built at Stanford. They introduced a new Layout-based Information Extraction Technique (LITE) and demonstrated its use automatically extracting semantic information from search forms and response pages. They also present results from experiments conducted to test and validate our techniques.

Michael Benedikt [3] verified about systems whose transitions consist of accesses to a Web-based data-source. An access is a lookup on a relation within a relational database, fixing values for a set of positions in the relation. AccLTL, is based on a first-order extension of linear-time temporal logic, interpreting access paths as sequences of relational structures. We also present a lower-level automaton model, Aautomata, which AccLTL specifications can compile into. They show that AccLTL and A-automata can express static analysis problems related to "querying with limited access patterns" that have been studied in the database literature in the past, such as whether an access is relevant to answering a query, and whether two queries are equivalent in the accessible data they can return. They proved decidability and complexity results for several restrictions and variants of AccLTL, and explain which properties of paths can be expressed in each restriction.

Jayant Madhavan et al [4] describe The Deep Web refers to content hidden behind HTML forms. In order to get to such content, a user has to perform a form submission with valid input values. The name Deep Web arises from the fact that such content was thought to be beyond the reach of search engines. The Deep Web is also believed to be the biggest source of structured data on the Web and hence accessing its contents has been a long standing challenge in the data management community .Over the past few years, they have built a system that exposed content from the Deep Web to web-search users of Google.com. The results of our surfacing are now shown in over 1000 web-search queries per-second, and the content surfaced is in over 45 languages and in hundreds of domains.

Bin He et al [5] describe the Internet– the Web has been rapidly "deepened" by massive databases online: While the surface Web has linked billions of static HTML pages, it is believed that a far more significant amount of information is "hidden" in the deep Web, behind the query forms of searchable databases. Static URL links– They are assembled into Web pages as responses to queries submitted through the "query interface" of an underlying database. Because current search engines cannot effectively "crawl" databases, such data is believed to be "invisible, "and thus remain largely "hidden" from users (thus often also referred to as the invisible or hidden Web.). Using overlap analysis between pairs of search

engines, a white paper estimated 43,000-96,000 “deep Web sites” and an informal estimate of 7,500 terabytes of data– 500 times larger than the surface Web. With its myriad databases and hidden content, this deep Web is an important yet largely-unexplored frontier for information search– While they have understood the surface Web relatively well, with various surveys. This article reports our survey of the deep Web, studying the scale, subject distribution, search-engine coverage, and other access characteristics of online databases. They noted that, while the 2000 study opens interest in this area, it focuses on only the scale aspect, and its result from overlap analysis tends to underestimate.

III. SYSTEM METHODOLOGY

A. Unbiased Estimation Algrithm

The Unbiased Estimation Algorithm (UEA) presents a solution for a novel problem: Aggregate Estimation for the Hidden Database with Checkbox Interface. In the hidden database with checkbox interface, a checkbox attribute is represented as a checkbox in the web interface. For example, in the home search website, features for a home are represented by checkboxes. The checkbox interface has its specialty. By checking the checkbox corresponding to a value v1, it ensures that all returned tuples contain the value v1. But it is impossible to enforce that no returned tuple contains v2 because unchecking v2 is interpreted as ”do-not-care” instead of ”not-containing-v ” in the interface. If one considers a feature of a products as a Boolean attribute, then the checkbox interface places a limitation that only TRUE, not FALSE, can be specified for the attribute also it is impossible to apply the existing techniques which require all values of an attribute to be specifiable through the input web interface. The unbiased-weighted-crawl which performs only random drill-downs on a novel structure of queries which refer to as a **left-deep tree**. The weight adjustment and low probability crawl for estimation accuracy.

Estimating the number of tuples in a hidden database is by itself an important problem. Many hidden databases today advertise their (large) sizes on public venues to attract customers. However, the accuracy of such a published size is not (yet) verifiable, and sometimes doubtful, as the hidden database owners have the incentive to exaggerate their sizes to attract access. Furthermore, many hidden databases, do not publicize their total sizes, while such information can be useful to the general public as an economic indicator for monitoring product growth. More generally, the ability to approximately answer aggregate queries can enable a wide range of third-party data analytics applications over hidden databases. For example, aggregates may reveal the quality, freshness, content bias and size of a hidden database, and can be used by third-party applications to preferentially select a hidden database with the best quality over other hidden databases.

However, applying capture-recapture over the existing sampling techniques for hidden databases leads to two problems, on estimation error and query cost, respectively. First, the estimations generated this way are biased and may have high variance

- Estimating the hidden database size, significant obstacles are present for estimating aggregates over a hidden database.
- The existing sampling-based techniques are not designed to answer aggregate queries, but to sample all tuples with equal probability. Thus, while these techniques may support an estimation of AVG queries, they cannot answer SUM or COUNT queries.
- Weight adjustment and low probability crawl is used for random drill-downs for estimation accuracy.
- The relative error occurred when the number of queries issued increases through the checkbox interfaces.
- The search results for the designated query are not controlled by the top-k restriction and it overflowing the search results.

B. Fast Moving Item Analysis (FMIA)

The proposed methodology overlapping the queries in a **left-deep-tree** data structure which imposes an order of all queries. Based on the order, it is capable of mapping each tuple in the hidden database to exactly one query in the tree, which is referred as the designated query. By performing a drill-down based sampling process over the tree and testing whether a sample query is the designated one for its returned tuple(s), it develops an aggregate estimation algorithm that provides completely unbiased estimates for **COUNT ,SUM and AVG** queries.

In addition through the research analysis of a **top-k restriction** on the number of returned tuples, and a limit on the number of queries one can issue through the web interface. Also, cache results of previous queries are maintained in web server space per **IP address per day.** The proposed system implement to ensure an unbiased estimation, the sum of weights of edges under one node should **equal to 1 and every edge** has a **non-zero weight** whenever there exists a tuple with designated query being a node in the subtree under this edge.

A **backtracking-enabled** random walk produces no bias, the variance of its estimation may be large when the underlying data distribution is highly skewed. The objective of weight adjustment is to reduce the estimation variance by “aligning” the selection probability of tuples in the database to the distribution of measure attribute (to be aggregated). For our purpose of estimating the database size, the measure attribute distribution is uniform (i.e., 1 for each tuple). Thus, adjust the transitional probability in the random walk based on the density distribution of “pilot” samples collected so far. After eight adjustment, each random walk produces an unbiased estimate with gradually reduced variance.

Divide-&-conquer is proposed to address this problem by carefully partitioning the database domain into a large number of subdomains, such that the vast majority of tuples belong to a small number of subdomains. Then, perform random walks over certain subdomains and combine the

results for estimation of the database size. The reduced size mismatch between the (sub-) query space and the database significantly reduces the final estimation variance, while only a small number of subdomains need to be measured, leading to very small increase on query cost.

In this research proposed methodology also concentrates in finding the most favorable **Fast Moving Item Analysis (FMIA)** that got sale and also got the order for the purchase of the particular item in the market. With these particular details it is easier for the producers in finding the more frequent item sale in particular area and the particular fast moving item easily from particular time to time.

- Cache results of previous queries are maintained in web server space and so eliminated the burden of database server using **tabu search algorithm**.
- Initiate the study of unbiased estimation of the size and other aggregates over a hidden database through its restrictive web interface.
- In this thesis implement a **backtracking-enabled** random walk technique to estimate hidden database size and other aggregates without bias.
- In this thesis implement a two additional techniques, **weight adjustment and divide-&-conquer**, to reduce the estimation variance.
- To combine the three techniques to produce **HD-UNBIASEDSIZE**, an efficient and unbiased estimator for the hidden database size. Similarly, propose **HD-UNBIASED-AGG** (AVG) which supports various aggregate functions and selection conditions.
- In this thesis provide a thorough theoretical analysis and experimental studies that demonstrate the effectiveness of proposed approach over real-world hidden databases.

IV.BACKTRACKING FOR CATEGORICAL DATA

First, Boolean attributes ensure that the sibling of an underflowing node always overflow. There is no such guarantee for categorical databases. Thus, to successfully backtrack from an underflowing branch, we must find one of its sibling branches that returns non-empty and count the number of such non-empty siblings (in order to compute p(q)). A Novel problem a non-empty branch always exists given an overflowing parent node. A simple backtracking approach is to query all branches to find the (COUNT of) non-empty ones, and then randomly choose a non-empty branch to follow.

- The other change required is the computation of p(q). If the above-mentioned simple backtracking is used, the computation of p(q) becomes p(q) = 1/ Qhi=1−1 ci, where ci is the number of non-underflowing branches for the i-th predicate then route to the top-valid query q.
- The two changes for categorical databases do not affect the unbiasedness of the estimation as the proof of Theorem 1 remains unchanged. These changes, however, do affect the query cost of the drill-down process.

In particular, if the above simple backtracking technique is used, we must issue queries corresponding to all branches to find the number of non-empty ones, leading to a high query cost for large-fanout attributes.

To reduce such a high query cost, to develop smart backtracking which aims to avoid testing all branches of a high fanout attribute. Consider a categorical attribute *Ai* with possible values v1, . . . , vw (w = |Dom(Ai)|). Assume a total order of the values which can be arbitrarily assigned for cardinal or ordinal attributes. In the following, we describe the random drill-down and the computation of $p(q)$ with smart backtracking, respectively.

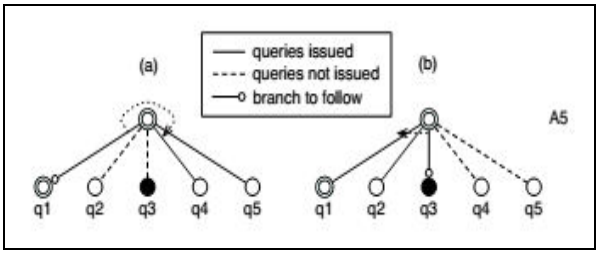

Figure 4.4: Example of Backtracking

V. DIVIDE AND CONQUER

Divide-&-conquer, a variance reduction technique which is independent of weight adjustment but can be used in combination with it. As mentioned in the introduction, divide-&-conquer provides the most significant variance reduction especially for the worst-case scenarios.

Divide-&-conquer is effective on reducing the estimation variance because it provides a significantly better alignment between the selection probability distribution for top-valid nodes and the measure attribute distribution. To understand why, consider a Boolean database with k = 1 and two top-valid nodes q and q0, at the second level (i.e., as a child of the root) and the bottom-level (i.e., n + 1-th level), respectively. Without divide-&-conquer, at the first drill-down, q has selection probability of 1/2 while q0 may have selection probability as small as p(q0) = 1/2n. This forms a striking contrast with the uniform distribution of the measure attribute (i.e., $| q| /m = | q0| /m = 1/m$ for each top-valid node), leading to a bad alignment between the two.

The total number of queries issued by the divide-&-conquer technique depends on the underlying data distribution. While theoretically a large number of queries may be issued, in practice the query cost is usually very small due to two reasons:

- One can see from that even a very small *r* can significantly improve the alignment and thereby reduce the estimation variance.
- As the experimental results show, for real-world hidden databases, even with a highly skewed distribution, the top-valid nodes are likely to reside on a small number of subtrees.

Furthermore, the following theoremshows that with the same query cost divide-&-conquer can significantly reduce the worst-case estimation variance. The random drill-down approach can also generate unbiased SUM and COUNT estimates for queries with conjunctive selection conditions. In particular, a conjunctive query can be considered as selecting a subtree which is defined with a subset of attributes (as levels) and, for each attribute involved, a subset of its values (as branches). The random drill-down approach can be applied to the subtree directly to generate unbiased estimations.

Algorithm: Back Tracking and Divide and Conquer
// Back Tracking LDT
1: $q \leftarrow$ root node. $p \leftarrow 1$. $i \leftarrow 1$.
2: Randomly generate $v \in \{0, 1\}$.
3: **Issue** $q\ 0 \leftarrow q \wedge (Ai = v)$. . for Step 1 (random drill-down)
4: **if** $q\ 0$ underflows **then**
5: $q \leftarrow q \wedge (Ai = 1 - v)$. Goto 2. . Backtracking
6: **else if** $q\ 0$ overflows **then** 7: **Issue** $q \wedge (Ai = 1 - v)$. . for Step 2 (computing $p(q)$)
8: **if** $q \wedge (Ai = 1 - v)$ is nonempty **then**
9: $p \leftarrow p/2$. . Update $p(q)$
10: **end if**
11: $q \leftarrow q\ 0$. $i \leftarrow i + 1$. Goto 2.
12: **end if**
13: **return** $m\tilde{} \leftarrow |\ q|\ /p$. .
14: Return an estimation for database size
// Divide and Conquer Algorithm LDT
for each column ci of pdb
if sum(ci) >= new_support
f1 = ii
else delete ci from pdb
for each row r
j of pdb
if sum(rj) < 2
delete r
j from pdb
for (k=2;| fk-1|>k-1;k++)
{
produce k-vectors combination for all columns of bdb;
for each k-vectors combination {ci1,ci2, ci3 … ,cik }
{
b= ci1 • ci2 •….•cik
if sum(b)>= new_support

A. Left Deep Tree Contruction

In this output form **(Fig 5.1)** is used to enabling the aggregate queries over a hidden database with checkbox interface by issuing a small number of queries (sampling) through its web interface. In this output form, an aggregate estimation algorithm is implemented and used that provides completely unbiased estimates for COUNT and SUM queries.

B. Aggregate Estimation with Fast Moving Item

In this form **(Fig 5.2)** aggregate estimation with fast moving item , in this form every node is corresponding to a query and a directed edge from a node to a child node indicates that the query corresponding to this child node includes all attributes item in the parent query and one additional attribute item.

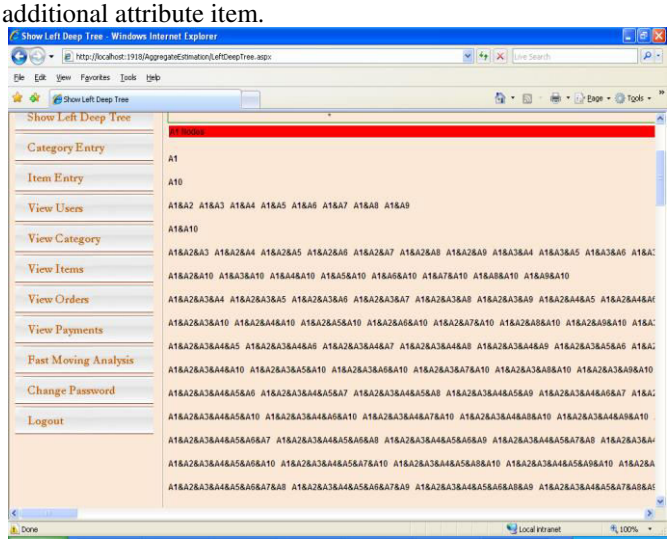

Fig 5.1 Left Deep Tree

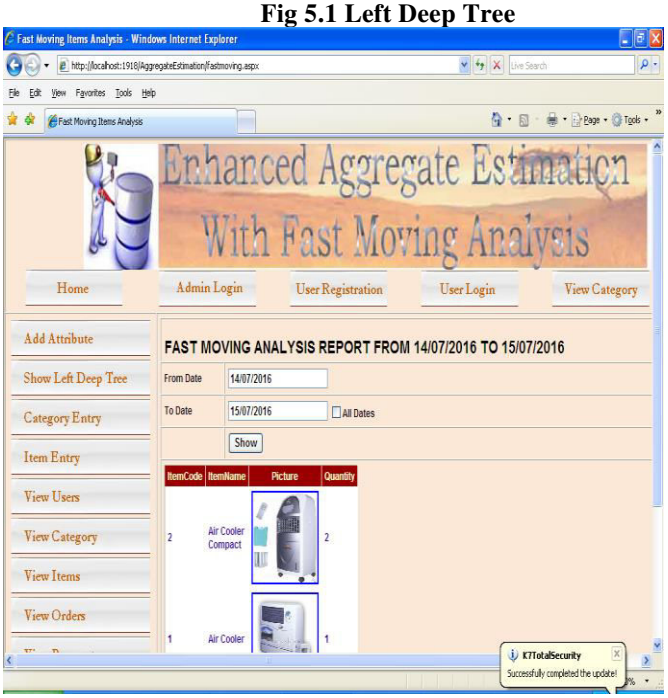

Fig 5.2 Fast Moving Analysis

VI.RESULTS AND DISCUSSION

The following table 6.1 shows the experimental results for Left Deep Tree and Fast moving Item base Left Deep Tree model. The table contains number of dataset query, average of left Deep Tree query and Average of Fast Moving item tree query. Find the aggregate estimation query is following formula estimated,

The following figure 6.1 shows the experimental results for Left Deep Tree and Fast moving Item base Left Deep Tree model. The figure contains number of dataset query, average of left Deep Tree query and Average of Fast Moving item tree query

$$\text{LDT} = {}^{qn}\textstyle\sum_{i=3} n/i \quad \text{FMILDT} = {}^{qn}\textstyle\sum 2^{*}\ n/3$$

Table 6.1 Aggregate Estimation Analysis: LDT-FMILDT

S.NO	Data set Query (n)	Left Deep Tree (LDT)	Fast Moving Item base Left Deep Tree (FMI-LDT)
1	100	54	13
2	200	113	24
3	300	152	25
4	400	206	26
5	500	315	31
6	600	387	32
7	700	419	30
8	800	498	32

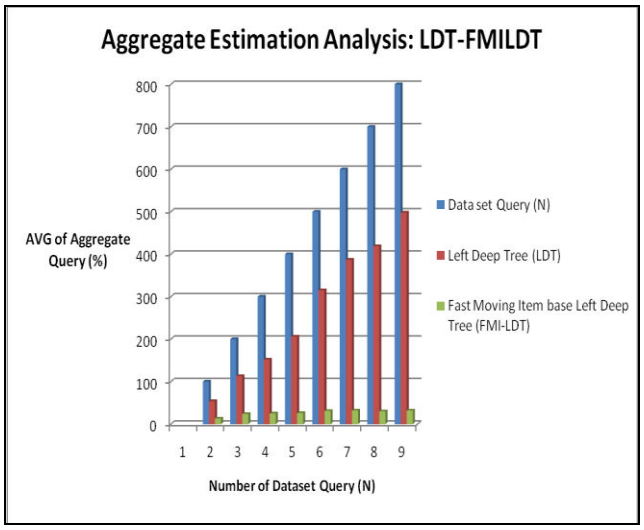

Fig 6.1: Aggregate Estimation Analysis: LDT-FMILDT

VII.CONCLUSION

In this paper addresses a novel problem where checkboxes exist in the web interface of a hidden database. To enable the approximation processing of aggregate queries and develops algorithm UNBIASED-WEIGHTED-CRAWL which performs random drill-downs on a novel structure of queries which we refer to as a left-deep tree and also propose weight adjustment and low probability crawl to improve estimation accuracy.

In this paper have initiated an investigation of the unbiased estimation of the size and other aggregates over hidden web databases through its restrictive web interface. We proposed backtrack-enabled random walk schemes over the query space to produce unbiased es timates for SUM, COUNT and AVG queries, including the database size. In this thesis fast moving item analysis system also described the two ideas, weight adjustment and capture & recapture, to reduce the estimation variance. The proposed system are provided theoretical analysis for estimation accuracy and query cost of the proposed ideas. The described a comprehensive set of experiments that demonstrate the effectiveness of proposed approach over synthetic and real-world hidden databases.

VIII.FUTURE ENHANCEMENTS

The paper has the scope for probing the hidden databases since query probing techniques have been widely used in the hidden database. The application become useful if the below enhancements are made in future. In this area, there are three key related subareas:

- Resource discovery, i.e., the discovery of hidden database URLs from the web,
- Interface understanding, i.e., the proper understanding of how to issue (supported) search queries through a web interface and to extract query answers from the returned web pages,
- Crawling, sampling and data analytics over hidden web databases, which is the most related to our problem.

REFERENCE

[1] C. Sheng, N. Zhang, Y. Tao, and X. Jin, ―Optimal algorithms for crawling a hidden database in the web,‖ Proc. VLDB Endowment, vol. 5, no. 11, pp. 1112–1123, 2012.

[2] Monster, Job search page [Online]. Available: http://jobsearch. monster.com/ AdvancedSearch.aspx, 2011.

[3] Epicurious, Food search page [Online]. Available: http://www.epicurious.com/ recipesmenus/advancedsearch, 2013.

[4] Homefinder, Home finder page [Online]. Available: http://www.homefinder.com/search, 2013.

[5] A. Dasgupta, X. Jin, B. Jewell, N. Zhang, and G. Das, Unbiased estimation of size and other aggregates over hidden web databases,‖ in Proc. Int. Conf.Manage. Data, 2010, pp. 855–866.

[6] M. Benedikt, G. Gottlob, and P. Senellart, ―Determining relevance of accesses at runtime,‖ in Proc. 30th ACM SIGMOD-SIGACT-SIGART Symp. Principles Database Syst., 2011, pp. 211–222.

[7] M. Benedikt, P. Bourhis, and C. Ley, ―Querying schemas with access restrictions,‖ Proc. VLDB Endowment, vol. 5, no. 7,pp. 634–645, 2012

[8] R. Khare, Y. An, and I.-Y. Song, ―Understanding deep web search interfaces: A survey,‖ ACM SIGMOD Rec., vol. 39, no. 1, pp. 33–40, 2010