Sampling From Databases Using B^+-Trees

Dimuthu Makawita*
Ngee Ann Polytechnic
Centre for Computer Studies
535, Clementi Road
Singapore 599489
mdp@np.edu.sg

Kian-Lee Tan
School of Computing
National Univ. of Singapore
3 Science Drive 2
Singapore 117543
tankl@comp.nus.edu.sg

Huan Liu
Dept. of Comp. Sc. & Eng.
Arizona State University
P.O. BOX 875406
Tempe, AZ 85287-5406
hliu@asu.edu

ABSTRACT

Sampling techniques are becoming increasingly important for large databases. However, the problem of obtaining a random sample from index structures has not received much attention. In this paper, we examine sampling techniques for B^+-tree. As the fanout of each node varies, a random walk through the index structure does not produce a good representative sample of the data set. We propose a new technique, called B^+-Tree based Weighted Random Sampling (BTWRS), that alters the inclusion probabilities of records accordingly to allow more records from leaves, along the paths with higher fanouts, to be extracted. We extensively evaluated our method, and the results show that BTWRS outperforms existing schemes in terms of the quality of the samples obtained and the efficiency of the sampling process. The proposed method can be readily adopted in existing commercial systems.

General Terms

Algorithms, Performance

Keywords

B^+-Tree, weighted random sampling, row sampling, page sampling, extent sampling, quality of samples

1. INTRODUCTION

On-Line Analytical Processing (OLAP) and Data Mining are tools for analyzing massive databases. These tools use a SQL database system to retrieve the required records for given queries. On a large database, the cost of executing such ad-hoc queries is expensive. In such cases, it is often reasonable (and desirable) to use a reduced amount of data or a subset of the available data to provide approximate answers to those obtained from the original data. Sampling techniques have been recognized as effective mechanisms that can produce a representative subset of the original data.

Existing statistical sampling methods [5] typically obtain samples from the data pages. This either requires scanning the database sequentially once or accessing sample data pages randomly. Both approaches are costly in terms of response time and resource consumption — the former needs to read the whole database; the latter incurs random I/O operations. Moreover, the representativeness of the samples depends on whether the data pages are clustered or not.

In this paper, we reexamine the sampling problem from a different perspective. Since most large databases are indexed, the index can be used to determine the samples to take. Because the index size is typically much smaller than the database, a method using the index should be more practical than one without using it. We consider in this work the B^+-tree as the index structure since it is available in almost all commercial database management systems (DBMS). The basic idea is to select random samples at the leaf level of the B^+-tree by traversing the tree from the root to leaves.

Though the concept is straightforward, the use of B^+-tree requires additional work to overcome the so called *structure-induced* problem. This is because some records get higher inclusion probabilities of being selected into the sample than others due to the structure of the tree. In other words, records from paths (i.e. from the root to a leaf node) with lower fanouts get selected more frequently than records from paths with higher fanouts. Such a sample is disproportionate in representing the original data.

In this paper, we propose a novel sampling scheme, in which the inclusion probability of a record depends on how dense the path of the record is. Therefore, a record will have a higher inclusion probability if it belongs to a denser path and vice versa. Therefore, a more representative sample can be obtained. The results of our experiments show that our method performs better, in terms of both quality of samples and efficiency, than the other methods discussed.

The rest of the report is organized as follows. In Section 2, we describe sampling from B^+-Trees and introduce our BTWRS method. We also discuss related work there. Section 3 describes our experimental setup/testbed and the parameters used to evaluate the quality of the samples. We

*The work was mostly carried out when the first author was at National University of Singapore.

discuss and analyze the experimental results in Section 4. Finally, Section 5 summarizes the conclusions drawn from our work and identifies and discusses the scope of future work. The notation used in paper are given in Table 1.

2. SAMPLING METHODS BASED ON B$^+$-TREES

We first introduce how a random sample can be obtained from a B$^+$-Tree by traversing the tree, and explain the B$^+$-Tree structure-induced problems. We then introduce Weighted Random Sampling (WRS) before we discuss our new sampling method BTWRS. We also review the related work, particularly the technique by Olken et al. [11].

2.1 Sampling from B$^+$-Trees by Random Walk from Root to Leaves

The simplest method that can be applied to sampling with B$^+$-Tree is to select a random path from the root to a leaf node and select a random record from that leaf node. The algorithm of selecting a sample of size s using this naive method of B$^+$-Tree based Simple Random Sampling (BT-SRS) is given in Figure 1.

1. Starting from the root node, at each internal node, select a branch at random
2. At the leaf level, select a record at random
3. Repeat the process s times from step 1

Figure 1: Algorithm of BTSRS

Suppose f_i is the fanout, which is the number of entries in a given node i, and the height of the tree is h. Then for each random walk from the root to a leaf node, it has to access exactly h nodes. The probability of selecting a child node (path) randomly from an internal node i, with a fanout f_i, is $\frac{1}{f_i}$. Note that at the leaf level we select a record randomly. Therefore, the probability of selecting a record k in a leaf node, denoted by B_k, is given by the formula,

$$B_k = \prod_{i \in Nodes_k} \left(\frac{1}{f_i}\right) \tag{1}$$

where $Nodes_k$ is the set of nodes the random walk finds from the root to the leaf node containing record k, i denotes the i^{th} node of $Nodes_k$, and $|Nodes_k| = h$. Now, this naive sampling method has a problem: records selected from a path of nodes with low fanouts will have a higher probability to be chosen into the sample than those from other paths of nodes with high fanouts. Clearly, such a sample is not a representative of the population. In the next section, we explain how our method can overcome this problem.

2.2 Weighted Random Sampling (WRS) with B$^+$-Trees

Method of WRS: Suppose it is required to draw a record r_k, out of N number of records, with the inclusion probability of record r_k being proportional to some weight W_k.

1. Calculate P_k for a record r_k
2. Generate another Uniform variate U_k such that $0 \le U_k \le 1$
3. If $P_k > U_k$ Then
   Accept r_k and include it into the sample
   Else
   Reject r_k
4. Repeat from Step 1 Until a sample of size s is obtained

Figure 2: Algorithm of WRS for selecting a sample of size s

Then the inclusion probability of r_k, denoted by P_k, is given by Equation (2), where W_{max} is the maximum of W_k for all k. Refer to [11, 19] for a detailed explanation of this method. The algorithm of selecting a sample of size s in this method is shown in Figure 2. Step 3 of Figure 2 is called as Acceptance/Rejection test.

$$P_k = \frac{W_k}{W_{max}} \tag{2}$$

(Note that $P_k \le 1$ since $W_k \le W_{max} \;\; \forall\, k$)

Intuitively, $W_{max} = \sum W_k$ so that $\sum P_k = 1$. Nevertheless, now $P_k << 1$ as $W_k << W_{max}$. Therefore, we expect this to incur a very high I/O cost due to the high rejection rate. In fact, we have proved this experimentally and it will be discussed later in the paper. Therefore, we use an alternative formula, which is also advocated by [11].

Now we describe our method of B$^+$-Tree based Weighted Random Sampling (BTWRS) as follows. Let B_k be the probability of selecting a record from a leaf by random walk from the root to a leaf. According to Equation (1), B_k can be written as,

$$B_k = \prod_{i \in Nodes_k} \left(\frac{1}{f_i}\right)$$

The value of B_k will be high for low f_is. But our approach is to get more records from the paths with higher fanouts. Therefore, we define W_k as given in Equation (3). Intuitively, now values of W_k will be high for low values of B_k for all k. In other words, now W_k will get higher values if the fanouts of the nodes belong to path k are high.

$$W_k = \frac{1}{B_k} \tag{3}$$

i.e., $W_k = \frac{1}{\prod_{i \in Nodes_k} \left(\frac{1}{f_i}\right)} \qquad \Longrightarrow W_k = \prod_{i \in Nodes_k} (f_i)$

Table 1: Notation used in the paper

Notation	Definition
Row_WRS	Weighted Random Sampling at Row Level
Row_SRS	Sampling by a Random Walk from Root to a Leaf at Row Level
Row_OLK	Sampling using Olken's method at Row Level
Pg_WRS	Weighted Random Sampling at Page Level
Pg_SRS	Sampling by a Random Walk from Root to a Leaf at Page Level
Pg_OLK	Sampling using Olken's method at Page Level
Xt_WRS	Weighted Random Sampling at Extent Level
Xt_SRS	Sampling by a Random Walk from Root to a Leaf at Extent Level
Xt_OLK	Sampling using Olken's method at Extent Level
#DV	Number of Distinct Values in Sample/Population
S	Sample Size
DS	Database Size
Z	Skewness Factor
D	Density of data (Proportion of Duplicates)
f_i	Actual Fanout of the node i
f_0	Fanout of the Root node
f_{max}	Maximum fanout of a node
f_{avg}	Average fanout of a node

We define $W_{max} = \max(W_k)$ and W_{max} is calculated as follows,

$$W_{max} = \max\left(\prod_{i \in Nodes_k} (f_i)\right)$$

$$\therefore W_{max} = (f_{max})^h$$

(Since $\max(f_i) = f_{max}$ *and* $|Nodes_k| = h$)

According to Equation (2),

$$P_k = \frac{W_k}{W_{max}} = (f_{max})^{-h} \cdot \left(\prod_{i \in Nodes_k} (f_i)\right) \qquad (4)$$

Therefore, according to Equation (4), P_k will be high for high f_is, and vice versa. This guarantees that records belonging to paths with high fanout nodes get more chances to be selected. We use the same algorithm given in Figure 1 to traverse from the root to a record in a leaf node. Now, a sample of size s is obtained based on our BTWRS method as shown in Figure 3.

We conducted experiments with $W_{max} = \sum W_k$ and the results showed that it only increased the I/O cost but the representativeness (quality) of the samples was not affected significantly (Refer to *#DV* and *I/O Cost* columns of Table 11). The higher I/O cost was due to the high rejection rate as $\sum W_k > (f_{max})^h$.

2.3 Related Work

The most related work in sampling from B$^+$-Trees is [11]. Olken et al. assume uniform inclusion probabilities for each record in the leaf nodes. Therefore, each record has the same probability being selected into the sample. But we claim here that just making each record to have the same fixed inclusion probability is inadequate, specially if this inclusion probability is very small. We elaborate on this problem with an example. Olken's fixed inclusion probability of records, P_k, as given in [11] is, $P_k = (f_0)^{-1} \cdot (f_{max})^{-h+1}$.

1. Starting from the root node, at each internal node select a branch at random until a leaf node reached
2. At the leaf level, select a record at random
3. Perform the Acceptance/Rejection test
4. Repeat from Step 1 until a sample of size s is obtained

Figure 3: Algorithm of BTWRS for selecting a sample of size s

Therefore, for a large fanout (e.g., 500) and even for a small h value (e.g., h=3), the value of P_k will be very low (e.g., $P_k = 10^{-8}$). Compare P_k with the probability, P_r, of a record being selected by a random walk from the root to a leaf node, where $P_r = (f_0)^{-1} \cdot (f_{avg})^{-h+1}$. Clearly $P_r > P_k$ as $f_{max} > f_{avg}$ (We ignore the rare extreme case of $f_{max} = f_{avg}$ which occurs only when all the nodes, other than the root, is completely filled with entries). Therefore, such a low probability, in fact, has no significant influence on selection of a record. Moreover, this low probability also incurs a high rejection rate, which leads to a high I/O cost that, in turn, leads to a poor performance.

The second main difference between the two methods is that the way of performing the Acceptance/Rejection test. Olken et al. performs this Acceptance/Rejection test at each node, other than the root, along a path from the root to a leaf. Therefore, a record can be rejected before the corresponding leaf is accessed. This method is called as 'Early Abort'. Performing this Acceptance/Rejection test at each node, one can reduce the number of I/Os, but it cannot improve the quality of samples. In our method, this Acceptance/Rejection test will be done only at the leaf level because we need to calculate the 'weight' of a record before performing the Acceptance/Rejection test.

3. EXPERIMENTAL DESIGN

We conducted the following experiments to compare our sampling method, in terms of efficiency and effectiveness, with the other methods discussed in our paper. For the purpose of the experiments, we generated data using the Zipf distributions [20, 14]. The *Skewness parameter* Z was varied. Although we studied several different values of Z varying from 0 to 4, we present our results only for values between 0 and 1, as for most of the practical situations Z gets values between 0 and 1. We varied the size of the table from 1 million to 20 million records. For each table size we studied the effect of different sample sizes of 1%, 10% and 20%. Sampling is done at three levels: *Record Level, Page Level* and *Extent Level*. At *Record Level*, the records are selected one at a time. Sampling at *Page Level* selects the entire number of records that are stored in the selected disk page. At *Extent Level*, all the records in eight consecutive pages, starting from the selected page, are taken into the sample. We also studied the effect of these three sampling levels in terms of IO cost and the quality of samples. In this paper, we used Sampling with Replacement (SWR) for simplicity and convenience. But one can easily replace SWR with Sampling without Replacement (SWOR) by removing duplicates using a data structure like a Hash table. We also consider the effect of the *Density Value, D*, which is the measure of the average number of duplicates in the data. D=0.0 implies that all values in a column are distinct while D=1.0 implies that all values in the column are identical. Experiments are carried out also to study the effect of duplicate values on the quality of samples, by varying D=0.2, D=0.5 and D=0.8. We evaluate our sampling method by using two measures: by the 'quality' of the sample and by the 'cost' incurred in getting the sample. We address these two measures below. The experiments were conducted on an Intel Pentium II 300MHz processor with 132MB RAM and 10.2GB Disk Drive.

1. By Quality of the Sample: The sample should be good enough to predict the characteristics of its population. The hard problem here is how to measure the 'goodness' or the 'quality' of the sample. We use two different methods to evaluate the quality of the samples. The first one is the estimation of number of distinct values of the population proposed by Chaudhuri et al. [3]. Suppose we pick a random sample of size s from a relation with N tuples. Let f_j, for $1 \leq j \leq s$, denote the number of distinct values, which occur exactly j times in the sample. Then the estimator, e, which estimates the number of distinct values, d, in the population, and the error metric, $rel\_error(e)$, used to measure the estimation error of d relative to N, are given by,

$$e = \sqrt{\tfrac{N}{r}} f_1^+ + \sum_{j=2}^{r} f_j \qquad \text{where} \qquad f_1^+ = \max\{f_1, 1\}$$

$$rel\_error(e) = \tfrac{d-e}{N}$$

The second method to evaluate the quality of a sample is *equi-depth* histograms [13, 15]. We use the samples obtained by the sampling methods discussed in this paper to build these histograms. Our intention is to find out which histogram gives the best estimation. The best histogram indicates the best sample, thus the best sampling method. Experiments are conducted using five different query sets as described in [15]. For a set of Q of N queries, the error E is computed as,

$$E = \tfrac{100}{N} \sum_{q \in Q} \tfrac{|S_q - S'_q|}{S_q}$$

where S_q and S'_q are the actual and the estimated size of the query results, respectively. We calculate E for each histogram constructed by each sample, given by each sampling method.

2. By Cost of getting the Sample: As the other measure to compare the samples, we use the I/O cost in getting a sample of a fixed size, for each sampling method. Here we ignore the cost of building the B^+-Tree and the cost of writing the results (samples), as it is the same for each sampling method for a given sample size.

4. ANALYSIS OF RESULTS

In this section, we present and analyze the results of the experiments described in the Section 3. The results of our experiments did not vary significantly in some different test cases, so we only present some. For example, as the skewness factor, (Z), and the Density value, (D), correlate, (i.e. when Z increases, D decreases and vice versa), some of the experimental results are omitted for some Z and D values. Due to the space limitation, we present the results of following arbitrarily selected test cases.

1. DS=10M, S=(1%,10%,20%), Z=1.0, D=0.1
2. DS=10M, S=(1%,10%,20%), Z=0.2, D=0.5
3. DS=20M, S=(1%,10%,20%), Z=0.5, D=0.8

The set of tables, from Table 2 to Table 4, shows the results based on the number of distinct values proposed by Chaudhuri et al. [3] as described in Section 3. The *Error* columns of these tables show that our sampling method gives the lowest error in estimating the number of distinct values. In other words, our method gives the best representative samples, at each level of sampling. The *I/OCost* columns of these tables show that the I/O costs for our sampling method, at each sampling level, are less than the I/O costs for the method proposed by Olken et al. Thus our sampling method is more efficient at each sampling level. These results also justify a few obvious facts such as the estimation error is reduced with the increase of the sample size and the larger the sample size, the higher the I/O cost.

The next set of tables, from Table 5 to Table 10, show the test results based on equi-depth histograms proposed in [13, 15] as described in Section 3. These results independently confirm that our sampling method gives the lowest error for each query type at each sampling level. In particular, our method gives very low errors at Row Level sampling.

Table 11 shows results based on the second formula used to calculate w_{max} as described in Section 2.2. In this case, the experimental results show that the quality of the samples do not affect significantly but the I/O cost increases (compare the $\#DV$ and $I/OCost$ columns of Table 3 and Table 11 for Row_WRS, Pg_WRS and Xt_WRS). Due to the space limitation, we only present the results shown in Table 11 for the second formula for w_{max}.

Figure 4 shows the relationship between the Sample Size and the Error for our sampling method and the method proposed by Olken et al, at page level. For this experiment, value Z is fixed at 0.5 and database size is 10M. It confirms the fact that the error reduces with the increase of the sample size. Figure 5 shows the relationship between Skew Factor Z and the Error for our sampling method and the method proposed by Olken et al, at page level. For this experiment, the database size is fixed at 10M and sample size is fixed at 10%. It shows the fact that when the Z is increased, the error is decreased. The reason could be: if the data is highly skewed (higher Z values), there will be many duplicate values in the data set (think about a situation where all the values in a data set are equal to one value, which is very unlikely to be occurred in real world). Thus in any sample, there will be more duplicates giving a good representative sample. For these two experiments, the error is calculated for the queries belong to *Set A* as defined in [15].

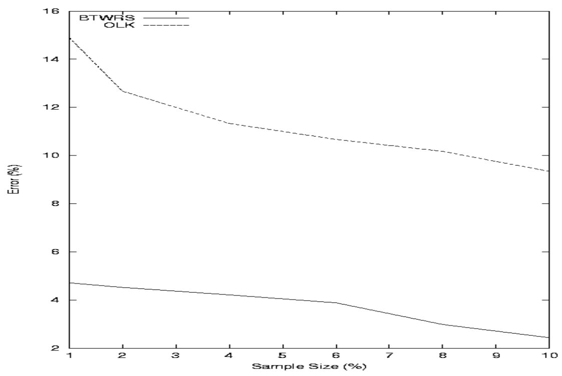

Figure 4: Sample Size Vs Error

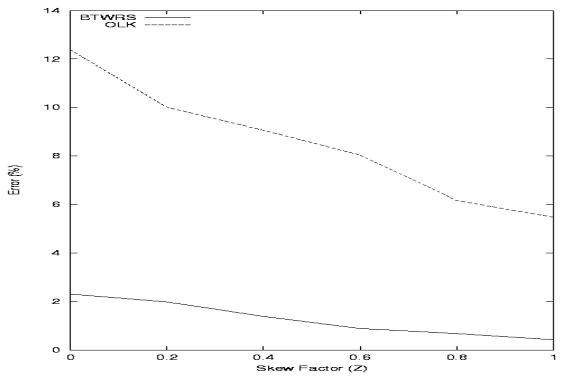

Figure 5: Skew Factor Vs Error

According to the results (I/O costs) of above experiments, Error versus I/O Cost can be summarized as given in Table 12 for all the sampling methods used. Therefore, Row Level sampling is the best method for the applications where high accuracy is needed, but may not be suitable for applications which need sampling frequently due to its high I/O cost. On the other hand, Page Level sampling is a good choice as both accuracy and I/O cost is quite acceptable. I/O cost is minimum for Extent Level sampling but there is no guarantee for a higher accuracy.

5. CONCLUSIONS

The experimental results show that the proposed BTWRS method gives the best results for all the test cases. Since this method uses the B$^+$-Tree without any modifications to its structure, it can be used with any database, which has a B$^+$-Tree index structure. Since the B$^+$-Tree is one of the most commonly used index structures, it is expected to be available with any DBMS. Thus our method could be readily adopted in existing commercial systems.

Future work could be to incorporate this BTWRS method as a function in DBMSs. Whenever a sample is needed from any file of a database, the DBMS would give the sample for a given size using BTWRS. Moreover, DBMSs can make use of samples of the databases to calculate/approximate the queries such as counts, averages, etc. Specially for the ad hoc queries and OLAP queries, the response time could be reduced greatly, if the DBMS uses a sample rather than going through the whole database. Here, Page Level sampling can be considered as a better choice since Row sampling is expensive and Extent sampling is low in accuracy. Another piece of future work would be to experiment the possibility of adopting our sampling method for index structures used for spatial and multidimensional data, for R trees and UB trees, for example.

6. ACKNOWLEDGEMENTS

We would like to thank Sukhamay Kundu for pointing out $W_{max} = \sum W_k$ for Equation (2) discussed in Section 2.2 and some constructive comments. We are also indebted to Hongjun Lu and Beng Chin Ooi for their valuable suggestions on the earlier versions of this work.

Table 2: Error and I/O Cost for Z=0.5, D=0.8 and Actual #DV = 16,000,000

Sampling Method	DS=20M								
	S=1%			S=10%			S=20%		
	#DV	Error	I/O Cost	#DV	Error	I/O Cost	#DV	Error	I/O Cost
Row_WRS	2,554,721	0.072	6,534,785	3,296,254	0.035	67,332,884	3,791,478	0.010	132,547,864
Row_OLK	2,047,229	0.098	612,354,913	3,008,872	0.050	60,912,453,972	3,454,781	0.027	11,956,485,219
Row_SRS	2,002,785	0.100	6,000,000	2,935,547	0.053	60,000,000	3,354,912	0.032	120,000,000
Pg_WRS	329,687	0.184	3,629	742,475	0.163	39,542	1,102,547	0.145	805,475
Pg_OLK	304,218	0.185	236,571	722,476	0.164	2,297,510	1,003,578	0.150	48,752,191
Pg_SRS	300,754	0.185	2,400	720,453	0.164	4,800	996,589	0.150	9,600
Xt_WRS	300,547	0.185	1,354	502,541	0.175	14,750	985,433	0.151	30,400
Xt_OLK	279,984	0.186	399,985	544,518	0.173	4,045,223	976,589	0.151	8,852,175
Xt_SRS	275,231	0.186	1,086	539,845	0.173	10,860	951,144	0.152	21,720

Table 3: Error and I/O Cost for Z=0.2, D=0.5 and Actual #DV = 500,000

Sampling Method	DS=10M								
	S=1%			S=10%			S=20%		
	#DV	Error	I/O Cost	#DV	Error	I/O Cost	#DV	Error	I/O Cost
Row_WRS	220,555	0.314	363,264	371,860	0.144	3,625,432	435,658	0.072	6,547,824
Row_OLK	207,068	0.329	26,833,260	356,650	0.161	267,644,040	420,841	0.089	495,473,642
Row_SRS	206,507	0.330	300,000	355,217	0.163	3,000,000	419,558	0.090	6,000,000
Pg_WRS	125,935	0.420	1,701	328,031	0.193	16,746	375,421	0.140	33,547
Pg_OLK	119,159	0.428	110,067	316,872	0.206	1,062,186	370,245	0.146	2,008,757
Pg_SRS	118,271	0.429	1,197	315,217	0.208	11,946	369,547	0.147	23,874
Xt_WRS	113,081	0.435	620	333,850	0.187	6,107	345,210	0.174	12,216
Xt_OLK	114,832	0.433	14,595	319,331	0.203	126,486	332,147	0.189	2,467,817
Xt_SRS	110,753	0.437	548	311,121	0.212	5,476	330,428	0.191	11,436

Table 4: Error and I/O Cost for Z=1.0, D=0.1 and Actual #DV = 1,000,000

Sampling Method	DS=10M								
	S=1%			S=10%			S=20%		
	#DV	Error	I/O Cost	#DV	Error	I/O Cost	#DV	Error	I/O Cost
Row_WRS	451,408	0.821	298,320	1,739,908	0.699	3,025,127	2,982,683	0.581	6,187,899
Row_OLK	421,442	0.824	33,153,195	1,699,563	0.703	312,188,739	2,753,473	0.603	657,835,972
Row_SRS	421,308	0.824	300,000	1,608,877	0.711	3,000,000	2,648,321	0.613	6,000,000
Pg_WRS	109,052	0.854	1,628	777,462	0.790	15,038	1,583,478	0.714	30,884
Pg_OLK	93,011	0.855	143,527	664,282	0.801	1,240,712	1,367,982	0.734	2,647,854
Pg_SRS	90,003	0.856	1,178	660,108	0.801	11,524	1,300,764	0.741	24,876
Xt_WRS	42,138	0.860	558	390,215	0.827	5,837	986,714	0.770	11,897
Xt_OLK	40,111	0.860	13,622	380,628	0.828	145,476	973,321	0.772	2,975,631
Xt_SRS	41,893	0.860	540	379,813	0.828	5,340	966,897	0.772	10,680

Table 5: Query Type A and S=10%

Sampling Method	Error(%)	I/O Cost
WRS_Row	0.034	275,042
OLK_Row	10.49	27,526,628
WRS_Page	3.49	1,499
OLK_Page	6.64	132,541
WRS_Xt	9.18	588
OLK_Xt	17.54	23,772

Table 6: Query Type A and S=1%

Sampling Method	Error(%)	I/O Cost
WRS_Row	1.99	30,661
OLK_Row	12.25	2,735,808
WRS_Page	5.26	133
OLK_Page	15.63	13,983
WRS_Xt	20.02	59
OLK_Xt	26.89	2,283

Table 7: Query Type B and S=10%

Sampling Method	Error(%)	I/O Cost
WRS_Row	0.20	274,772
OLK_Row	8.15	27,237,657
WRS_Page	4.10	1,508
OLK_Page	7.19	109,929
WRS_Xt	5.84	590
OLK_Xt	10.84	14,781

Table 8: Query Type C and S=10%

Sampling Method	Error(%)	I/O Cost
WRS_Row	0.97	275,611
OLK_Row	13.47	27,367,200
WRS_Page	10.90	1,518
OLK_Page	17.70	111,921
WRS_Xt	11.13	601
OLK_Xt	24.16	12,915

Table 9: Query Type D and S=10%

Sampling Method	Error(%)	I/O Cost
WRS_Row	0.08	275,002
OLK_Row	17.33	27,147,699
WRS_Page	12.83	1,498
OLK_Page	28.33	113,502
WRS_Xt	18.00	588
OLK_Xt	34.75	15,795

Table 10: Query Type E and S=10%

Sampling Method	Error(%)	I/O Cost
WRS_Row	0.28	274,992
OLK_Row	13.34	27,109,212
WRS_Page	7.38	1,516
OLK_Page	14.40	114,021
WRS_Xt	18.42	592
OLK_Xt	28.31	14,727

Table 11: Error and I/O Cost for Z=0.2, D=0.5 and Actual #DV = 500,000 (for $w_{max} = \sum W_k$)

Sampling Method	DS=10M								
	S=1%			S=10%			S=20%		
	#DV	Error	I/O Cost	#DV	Error	I/O Cost	#DV	Error	I/O Cost
Row_WRS	219,903	0.315	4,793,819	370,732	0.145	47,962,513	436,921	0.071	95,925,531
Row_SRS	206,507	0.330	300,000	355,217	0.163	3,000,000	419,558	0.090	6,000,000
Pg_WRS	125,198	0.421	24,085	328,998	0.192	239,602	374,648	0.141	478,828
Pg_SRS	118,271	0.429	1,197	315,217	0.208	11,946	369,547	0.147	23,874
Xt_WRS	115,886	0.432	9,274	332,771	0.188	93,977	346,012	0.173	187,882
Xt_SRS	110,753	0.437	548	311,121	0.212	5,476	330,428	0.191	11,436

Table 12: Error and I/O Cost

Sampling Level	Error	I/O Cost
Row	Low	High
Page	Medium	Medium
Extent	High	Low

7. REFERENCES

[1] S. Acharya, P. B. Gibbsons, V. Poosala, and S. Ramaswamy. The aqua approximate query answering system. *Proceedings of the ACM SIGMOD*, pages 574–576, 1999.

[2] S. Acharya, P. B. Gibbsons, V. Poosala, and S. Ramaswamy. Join synopses for approximate query answering. *Proceedings of the ACM SIGMOD*, pages 275–286, June 1999.

[3] S. Chaudhuri, R. Motwani, and V. Narasayya. Random sampling for histograms construction: How much is enough? *Proceedings of the ACM SIGMOD*, pages 263–274, June 1998.

[4] S. Chaudhuri, R. Motwani, and V. Narasayya. On random sampling over joins. *Proceedings of the ACM SIGMOD*, pages 263–274, June 1999.

[5] W. G. Cochran. *Sampling Techniques.* Wiley, 1997.

[6] J. Ernvall and O. Nevalainen. An algorithm for unbiased random sampling. *The Computer Journal*, 1(25), 1982.

[7] C. Harris-Jones and T. L. Haines. Sample size and misclassification: Is more always better? Working Paper AMSCAT-WP-97-118, AMS Center for Advance Technologies, December 1997.

[8] C. C. Low. The influence of skew distributions on query optimization. Master's thesis, Department of Computer Science, Royal Melbourne Institute of Technology, March 1999.

[9] F. Olken. *Random Sampling from Databases.* PhD thesis, Department of Computer Science, U.C., Berkeley, 1993.

[10] F. Olken and D. Rotem. Simple random sampling from relational databases. *12th International Conference on Very Large Databases (VLDB)*, pages 160–169, 1986.

[11] F. Olken and D. Rotem. Simple random sampling from relational databases. *12th International Conference on Very Large Databases (VLDB)*, pages 269–277, 1989.

[12] P. Palvia. Expressions for batched searching of sequential and hierarchical files. *ACM Transactions on Database Systems*, 10(1):97–106, 1985.

[13] G. Piatetsky-Shapiro and C. Connell. Accurate estimate of the number of tuples satisfying a condition. *ACM SIGMOD Conference*, pages 256–276, 1984.

[14] V. Poosala. Zipf's law. Technical Report 900 839 0750, Bell Laboratories, USA, 1997.

[15] V. Poosala, Y. E. Ioannidis, P. J. Hass, and E. J. Shekita. Improved hiistograms for selectivity estimation of range predicates. *ACM SIGMOD Conference*, pages 294–305, June 1996.

[16] F. Provost, D. Jensen, and T. Oates. Efficient progressive sampling. *Fifth International Conference on Knowledge Discovery and Data Mining*, June 1999.

[17] R. Ramakrishnan. *Database Management Systems.* McGraw-Hill, 1997.

[18] A. M. Tenenbaum and M. J. Augenstein. *Data Structures Using Pascal.* Prentice-Hall International, Inc., second edition, 1986.

[19] C. K. Wong and M. C. Easton. An efficient method for weighted sampling without replacement. *SIAM Journal on Computing*, 9(1):111–113, February 1980.

[20] G. K. Zipf. Human behavior and the principle of least effort. Technical report, Addison-Wesley, Reading, MA, 1949.