COPYRIGHT © 2010 JCIT, ISSN 2078-5828 (PRINT), ISSN 2218-5224 (ONLINE), VOLUME 01, ISSUE 01, MANUSCRIPT CODE: 100711

Query Optimization on Relational Databases for Supporting Top-k Query Processing Techniques

M.A. Kashem, Abu Sayed Chowdhury, Rupam Deb, and Moslema Jahan

Abstract—Information systems apply various techniques to rank query answers. Ranking queries (or top-k queries) are dominant in many emerging applications, e.g., similarity queries in multimedia databases, searching web databases, midlewares and data mining. In such application domains, end-users are more interested in the most important (top-*k*) query answers in the potentially huge answer space. Thus for why in relations' join, the suitable size of relations inputs for getting top K query answers must be determined. This paper describes an algorithm for finding input size of N relations in rank aware queries to efficiently answer to the queries with join of N relations for getting top K query answers and it is observed from the experimental result that the time of query processing extraordinarily will be decreased.

Index Terms—Relational database, query optimization, top-k, Information retrieval, join of N relations.

1 INTRODUCTION

Emerging applications that depend on ranking queries warrant efficient support of ranking in database management systems. Integration of database and information retrieval (IR) technologies has been an active research topic recently [11]. Top-k queries aim at providing only the top-k query results, according to a user-specified ranking function. The increasing importance of top-k queries warrant an efficient support of ranking in the relational database management system (RDBMS) and has recently gained the attention of the research community. Supporting ranking gives database systems the ability to efficiently answer IR queries. For many years, combining the advantages of databases and information retrieval systems has been the goal of many researchers. An important subject in this connection is determining the suitable size of inputs in relation N for answering to the rank aware query so that in this manner top-k query answers are gotten. It is vital in information integration with large scale to select top-k rank aware from multiple sources and it has also basic role for minimizing transfer cost because as the size of relations become smaller, transfer cost become less. For answering to a query Efficient algorithms have been developed for answering to ranking queries in middleware environment [1]. The other algorithm that is used for answering to the ranking queries is estimating input sizes by means of statistical relations, monotony hypothesis and random variables [3, 4]. The other new innovation is making ranking laws in relational databases [5]. The other solution in this connection is improving the join and using the ripple join that minimize the time in order to get estimation with relatively acceptable precision for query results. The main idea of ripple join is join algorithm aware the suggested rank for supporting join queries with top K relational database [6]. But there are the other methods for answering to the queries with top-k, which are getting top-k by changing query optimization theorem to the aware searches [7].

In this paper, we provide an idea of efficient pruning of non requisite records for supporting top k queries with join of N relation.

- *Faculty, Department of Computer Science and Engineering, Dhaka University of Engineering and Technology (DUET), Gazipur.*
- *E-mail:*
- *{drkashemII, kayes04_duet, rupam_duet, semi_cse2k3}@yahoo.com*

2 QUERY OPTIMIZATION BY RANKING QUIRES ON RELATIONAL DATABASE

2.1 Relational Database

A relational database is a collection of data items organized as a set of formally described tables from which data can be accessed or reassembled in many different ways without having to reorganize the database tables. Fig.1 shows a relationship of student database.

2.2 Conventional TOP-K Query Processing

Top-k processing connects too many database research areas including query optimization, indexing methods and query languages. As a consequence, the impact of efficient top-k processing is becoming evident in an increasing number of applications. The following examples illustrate real-world scenarios where efficient top-k processing is crucial. The

examples highlight the importance of adopting

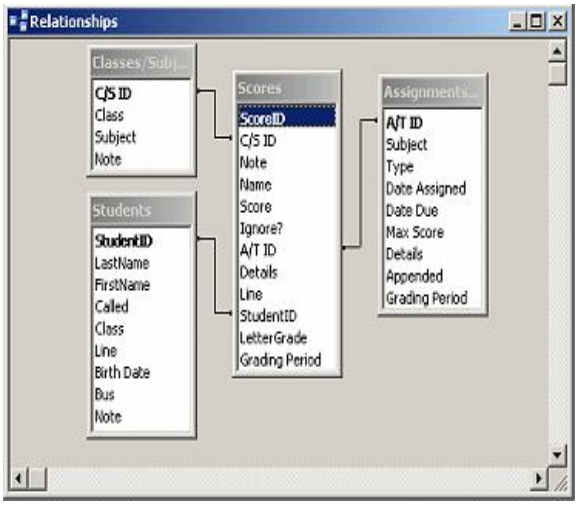

Fig.1: A Relational Database

efficient top-k processing techniques in traditional database environments. Consider a user interested in finding a location (e.g., city) where the combined cost of buying a house and paying school tuition for 5 years at that location is minimum. The user is interested in the four least expensive places. Assume that there are two external sources (databases), Houses and Schools, that can provide information on houses and schools, respectively.

HID	Location	Price
1	Mirpur	9000
2	Badda	10000
3	Farmgate	10500
4	Motijhil	11000
5	Uttara	12500
6	Dhanmondi	14000

House

SID	Location	Tution
1	Farmgate	3000
2	Badda	3500
3	Mirpur	4000
4	Mirpur	4500

School

HID	SID	Price+3*Tution
3	1	25500
2	2	27500
1	3	29000
1	4	31500

Join Result

Fig.2: A top k query example

The houses database provides a ranked list of the cheapest houses and their locations. Similarly, the Schools database provides a ranked list of the least expensive schools and their locations. The search must be done in two relations of house and school in the database, according to the following query as shown in fig.2:

```
SELECT h.HID, s.SID, h.Price+3*s.Tuition as Cost
FROM House h,School s
WHERE h.Location=s.Location
ORDER BY Cost
LIMIT 4
```

In current database systems, the queries in the example are evaluated as follows:

- First, the input tables are joined according to the join condition.
- Then, for each join result, the global score is computed according to the given function.
- Finally, the results are sorted on the computed combined score to produce the top-k results and the rest of the results are dropped.

Here two major expensive operations are involved:

- Joining the individual inputs and
- Sorting the join results.

Here the top four results cannot be returned to the user until all join results are generated .For large numbers of co-located houses and schools, the processing of such query in the traditional manner is very expensive as it requires expensive join and sort operation for larger amounts of data.

2.3 Query optimization

The objective in query optimization is to select an efficient execution strategy. The query optimization process shown in fig.3 consists of getting a query on N relations and generating the best query execution plan (QEP). For a given query, the search space can be defined as the set of set of equivalent operator trees that can be produced using transformation rules.

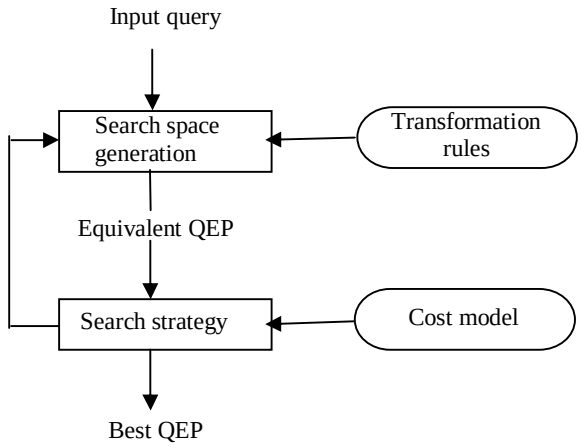

Fig.3: Query optimization process

Regarding different search spaces there would be different shape of the join tree as shown in fig. 4. [10]

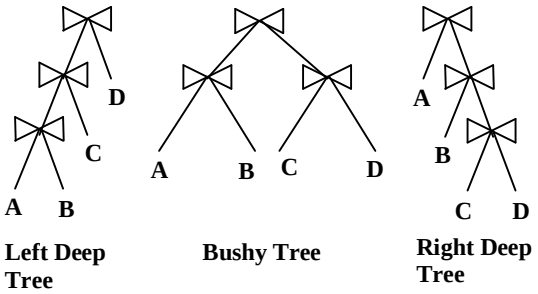

Fig.4: Types of join tree

2.4 Ranking Queries

In rank aware query(also known as top-k query), query define on M attribute A_1, A_2,..., A_N and relation N in the form of R_1, R_2,...,R_N that each Ai (i=1:M) belongs to one relation Rj (j=1:N). Each of the attributes have special domain in comparison with their kind. According to the query, a series of attributes of these relations are applied for projection, a series of attributes of these relations are used for restriction and join. In the rank aware queries there is a part for ranking that some of relations attributes are presented in the form of a ranking relation which is called ranking function. Ranking function f is formed in the form of attribute M' that is M' <=M. A theory that we have for ranking function f is this: ranking function changes in comparison with all relations are monotonic. In addition to this, the number of suitable answers in rank aware queries is determined too that is just Top K. Consider a set of relations R1 to Rm. Each tuple in Ri is associated with some score that gives it a rank within Ri. The top-k join query joins R1 to Rm and produces the results ranked on a total score. The total score is computed according to some function, say F that combines individual scores. Note that the score attached with each relation can be the value of one attribute or a value computed using a predicate on a subset of its attributes.

A possible SQL-like notation for expressing a top-k join query is as follows:

SELECT *

FROM R_1, R_2, . . . ,Rm

WHERE join condition (R_1, R_2, . . . ,Rm)

ORDER BY F (R_1.score, R2.score, . . . , Rm.score)

LIMIT k;

Where LIMIT limits the number of results reported to the user.

2.5 TOP-K Query Processing Techniques

The classification of top-k query processing techniques based on multiple designs is shown in fig.5. [12]

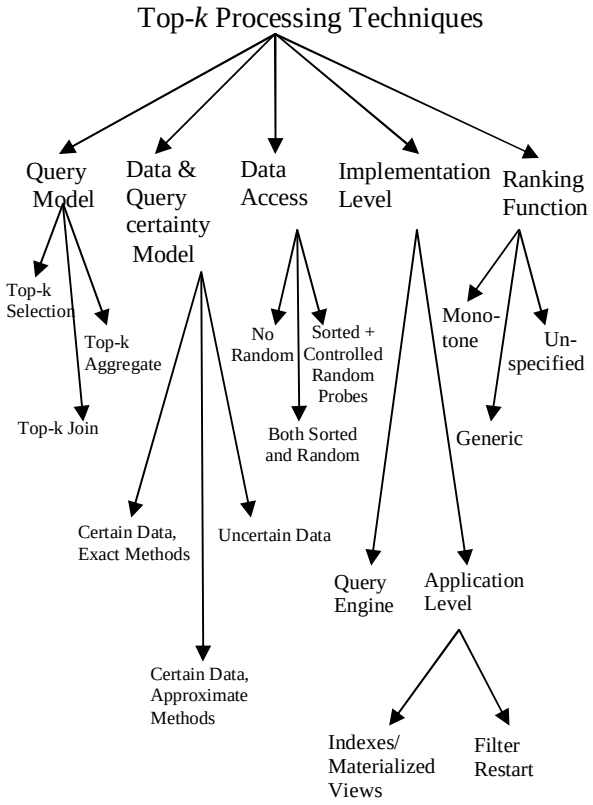

Fig.5: Classification of top-k query processing techniques.

Top-k processing techniques are classified based on the restrictions they impose on the underlying ranking (scoring) function. Most proposed techniques assume monotone scoring functions. Few proposal address general functions.

3 Analysis of Query Processing

3.1 Supporting TOP-K Queries

For answering to a query with Top K the traditional solution is to do the join on N relations firstly and then sort the answers according to ranking function and select Top k. This algorithm with a view to implementation is very simple but in fact for getting Top k most of the tuples are not important and they have not any role in the ultimate answer, these extra tuples must be pruned according to correct strategy. A key property of top-k queries is that users are interested only in the first k results and not in a total ranking of all query results. This property directly impacts the optimization of top-k queries by optimizing for the first k results. Traditionally, most real-world database systems offer the feature of First-N-Rows-Optimization. Users can turn on this feature when desiring fast response time to receive results as soon as they are generated. This feature translates into respecting the “pipelining” of a plan as a physical plan

property. For example, for two plans P1 and P2 with the same physical properties, if P1 is a pipelined plan (e.g., nested-loops join plan) and P2 is a non-pipelined plan (e.g., sort-merge join plan), P1 cannot be pruned in favor of P2, even if P2 is cheaper than P1.

3.1.1 Pruning Inputs

Pruning inputs for supporting Top K in queries with join of N relations, input parameters in company with relation N is in the form of R1,R2,…,RN that each Ri (i=1:N) containing some attributes for joining with other relation that is presented in this form Ri.Join_Feilds[j] (j=1:N-1). Also there is a ranking function f with monotonic increase that is conformable with equation 1.

$f(R_2.Rank\_Feilds[l_1], R_2.Rank\_Feilds[l_2], \ldots\ldots\ldots\ldots, R_n.Rank\_Feilds[l_n])$ ----------(1)

3.2 Specifying Input Size

According to the query, a series of attributes of these relations are applied for projection, a series of attributes of these relations are used for restriction and join. in this manner:

Projection- R[].Projection_Feilds[j'],
Restriction- Ri.Restriction_Feilds[j"]

Which are applied in the general algorithm.

3.2.1 Input Size for Two Relations

Consider tow relations R1 and R2 that are arranged according to their ranking attributes in decreasing manner. The flow chart for input size specification is shown in fig.6. If records of R1 end but the number of records that was the join condition between two relations become less than K it means that the ultimate answer, which is gotten from join of two relations, is less than K. we also do these stages in the same way for R2 and record the results.

3.2.2 Concept for N Relations' Join

The important fact to join N relation is the order of joins action between relations; it means that which relations must be joined firstly so that the number of operations becomes minimum, for this purpose join tree must be made. The most important point for determining suitable input size is determining suitable place for each relation in the join tree as shown in fig.2. We can get the best state by means to bushy tree which is used in implementation.
Suggested idea for N relations is stated below:

A. Calculate the input size for every relations of R[] .
B. Place the two relation of least input size, one (that is minimum between two) to the left sub-tree as the parent of left sub-tree and the other on the right sub-tree as parent of right sub-tree.
C. The child of left sub-tree and right sub-tree are constructed in similar manner as mentioned in step b for rest of the relations.

3.3 Query Analyzer

Fig. 7 shows processing of query analyzer. The required attributes are selected from each of the relations according to the query and rest of

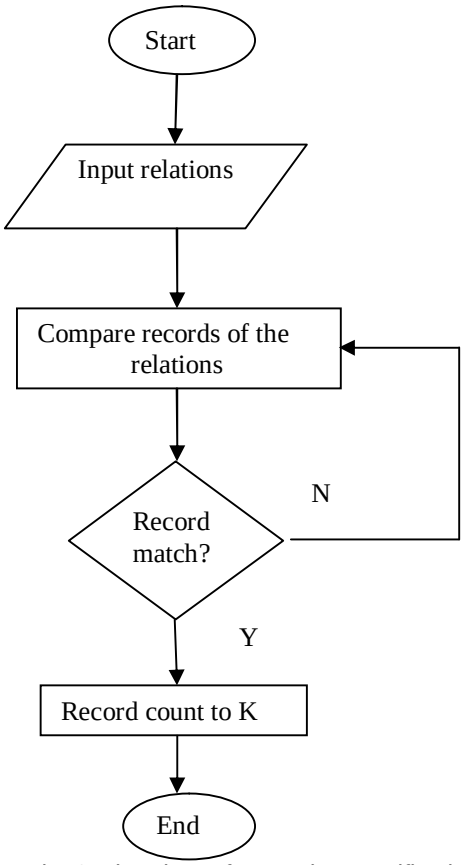

Fig. 6: Flowchart of Input size specification

attributes is pruned. Then the size of relations is calculated. By forming bushy tree order of join action is evaluated. Finally join between tables is done by pruning records to get top-k query answers.

3.4 Algorithm for TOP-K Queries

```
Algorithm TOPk_queries ( R)
{
// Array R[] holds the relations.
// Relations are ordered based on ranking function.
select requisite attributes by projection and restriction;
for ( i=1; i<=n; i++)
  for ( j=1;j<=N;j++)
     if ( i ≠ j)
       {
     Calculate input size of R[i] and R[j] ;
     Construct bushy tree for all relations of R[]
     based on input size and ranking function;
```

}

Create join between relations based on constructed tree;

Write join result;

}

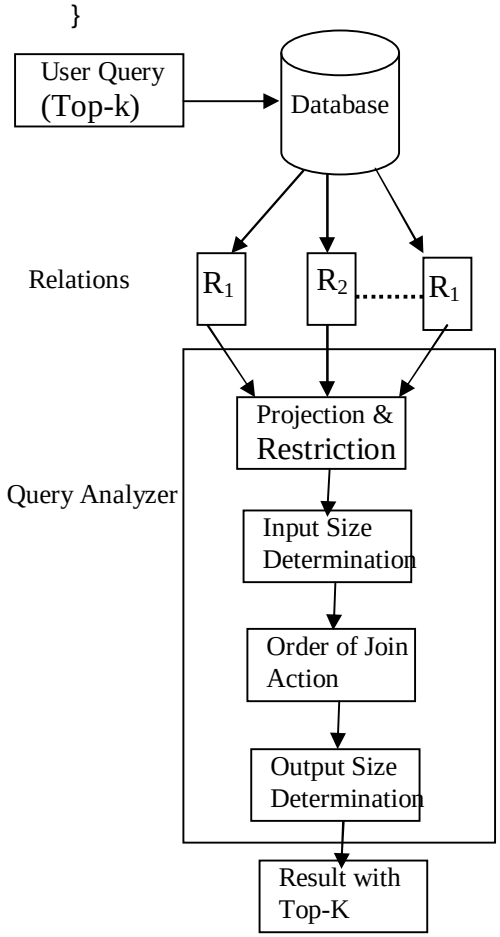

Fig.7: Processing the query in Query Analyzer.

4 EXPERIMENTAL RESULTS

The structure of the Top K queries for N relation join have been implemented over Visual Studio.Net & PostgreSQL database running on 2.66GHz Pentium IV processor with 512MB memory. The operating system is Microsoft XP. The results are compared on the same hardware configuration. We use a synthetic data set of three database (A, B,C). Table A and table B each have one boolean attribute with 0.4 as their selectivities. The three tables have 2, 2, 1 ranking predicates respectively. The ranking predicated have the same cost. Scores of different ranking predicated are within the range between 0 and 1 which are generated by different distribution including uniform, normal and cosine distribution. Each table has two attributes jc1 and jc2 as join columns. We use a simple top-k query Q where summation is used as scoring function.

Q =

SELECT *

FROM A, B, C

WHERE A.jc1= B.jc1 AND B.jc2 = C.jc2 AND A.b and B.b

ORDER BY f1(A.p1) + f2(A.p2)+ f3(B.p1) + f4 (b.p2) + f5(C.p1)

LIMIT k

Query Optimization implemented by using pruning architecture reduces the query process time drastically and hence reduces cost. Different execution time for different values of K the comparison of time cost for traditional and suggested systems shows in figure 8. We also analyze the accuracy of information which is gotten by means of this approach, in all of the analyzed queries and for different Ks, the answers of this approach is100% equal to answers that is gotten by means of traditional approaches.

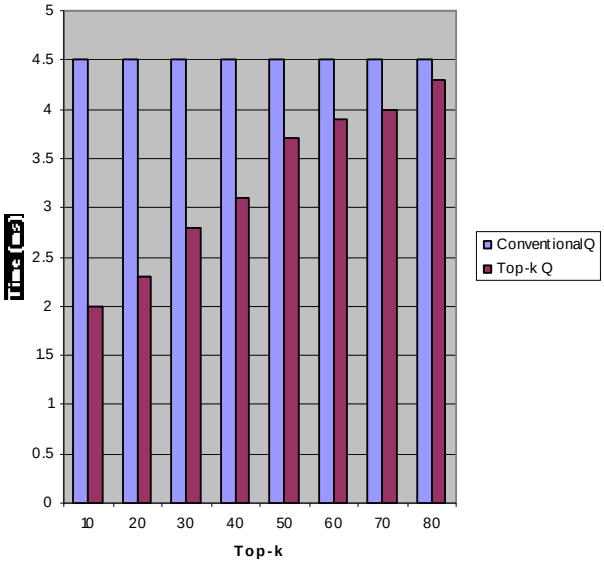

Fig.8: Comparison of time cost for conventional and proposed systems

5 CONCLUSION

Query optimization on relational database has contributed to the existing system for the query operations significantly. It is essential for large information retrieval system. The presented approach in big databases that has high information volume and their aim is implementing Rank aware Queries with small Ks amount is suitable and efficient. It is also efficient for systems that their aim is information integration from some systems proportionate to their requisite queries and relations because it optimizes information volume of relations proportionate to query and determination of requisite input size of relations. This approach makes time cost less from 40 to 50% for small Ks and also makes requisite volume of relation less for transferring from half of the primary volume. As the database is growing gradually the retrieval time is also increased in

exponential order. We have implemented the query access system within single processor system and also presented how we can perform ranking operation on the result of query operation and make decision.

REFERENCES

[1] Fagin,R. and A. Lotem, 2001.Optimal Aggregation Algorithms for Middleware.In Proceedings of PODS 2001, Santa Barbara, USA.

[2] Marian, A. and N. Bruno, 2004. Evaluating Top-K Queries over Web Accessible Databases. ACM Transactions on Database Systems, 29 (2): 319-362.

[3] Ilyas, I.F. and W.G. Aref, 2006. Adaptive Rank aware Query Optimization in Relational Databases. ACM Transactions on Database Systems.

[4] Ilyas, I.F. and R. Shah, 2004. Rank-aware Query Optimization.SIGMOD2004, June13-18, Paris, France.

[5] Ilyas, I.F. and C. Li, 2005. Ranksql: Query algebra and optimization for relational top-k queries.SIGMOD 2005, June 1416, Baltimore, Maryland, USA.

[6] Ilyas, I.F. and W.G. Aref, 2003. Supporting Top-k Join Queries in Relational Databases. Proceedings of the 29th VLDB Conference, Berlin, Germany.

[7] Zhang, Z. and S. Hwang, 2006. Boolean+Ranking Querying a Database by KConstrained Optimization.SIGMOD 2006, June 27.29, 2006, Chicago, Illinois, USA.

[8] Liu Jie and Liang Feng, 2006. A Pruning-based Approach for Supporting Top-K Join Queries.ACM, Edinburgh, Scotland.

[9] Reza Ghaemi, Amin Milani Fard, Hamid Tabatabaee, and Mahdi Sadeghizadeh September 2008, Evolutionary Query Optimization for Heterogeneous Distributed Database Systems. World Academy of Science.

[10] H.Shirgahi, H. Motameni, A.H. Gandomi and P. Valipour, 2008. A New Approach of Query Optimization with Join of N Relations, World Applied Sciences Journal.

[11] S. Chaudhury, R. Ramakrishnan, and G. Weikum. Integrating db and ir technologies, CIDR, Pages 1-12, 2005.

[12] Ilyas, I.F, Beskales, G, Soliman M. A, 2008, A survey of top-k query processing techniques in relational database systems. ACM Comput. Surv. 40, 4, Article 11 (October 2008).

Dr. Mohammod Abul Kashem completed Ph.D from national university of Lviv Polytechnic, Ukraine in 2001. He is a member of IEEE, IEB-Bangladesh and serving as an associate professor and Head of the Computer Science and Engineering department in Dhaka University of Engineering and Technology, Gazipur-1700, Bangladesh. He is keen on research fields-Image Processing, Signal Processing and database. His E-mail is drkashem11@yahoo.com.

Abu Sayed Chowdhury who born in Tongi, Gazipur, Bangladesh, has completed his B.Sc. Engineering degree from Computer Science and Engineering (CSE) department of Dhaka University of Engineering & Technology (DUET), Gazipur, Bangladesh. He has interest in Distributed Search Techniques and Advanced Database Systems. He is a lecturer in Computer Science and Engineering dept. in Dhaka University of Engineering & Technology, Gazipur, Dhaka, Bangladesh. Mr. Chowdhury is now Associate Member of the Engineers Institution, Bangladesh (IEB). Email address: kayes04_duet@yahoo.com.

Rupam Deb who born in Hathazari, Chittagong, Bangladesh, has completed his B.Sc. Engineering degree from Computer Science and Engineering (CSE) department of Dhaka University of Engineering & Technology (DUET), Gazipur, Bangladesh. He has interest in Image processing and Software Engineering. He is an Assistant Professor in Computer Science and Engineering dept. in Dhaka University of Engineering & Technology, Gazipur, Dhaka, Bangladesh. Mr. Deb is now Member of the Engineers Institution, Bangladesh (IEB). Email address: rupam_duet@yahoo.com

Moslema Jahan who born in Kahaloo, Bogra, Bangladesh, has completed his B.Sc. Engineering degree from Computer Science and Engineering (CSE) Department of khulna University of Engineering & Technology (KUET), Khulna, Bangladesh. She has interest in Parallel and Distributed Computing and Advanced Database System. She is a lecturer in Computer Science and Engineering dept. in Dhaka University of Engineering & Technology, Gazipur, Dhaka, Bangladesh. Miss. Jahan is now Associate Member of the Engineers Institution, Bangladesh (IEB). Email address: semi_cse2k3@yahoo.com.