Selectivity Evaluation in Distributed Database Query Operations: Static vs Dynamic Techniques

Surbhi Bansal
Research Scholar
Department of Computer Science and Engineering
Guru Nanak Dev University, Amritsar, Punjab.

Sofia Gupta
Research Scholar
Department of Computer science and engineering
Guru Nanak Dev University, Amritsar, Punjab.

Rajinder Singh Virk
Assistant Professor
Department of Computer science and engineering
Guru Nanak Dev University, Amritsar, Punjab.

ABSTRACT
In distributed database query optimization, one of the main factors affecting the performance of an execution strategy is the intermediate fragment sizes produced during the execution of the sub query operations. This paper analyses static vs. dynamic calculation for selectivity of intermediate relations generated in query processing. A Dynamic model for selectivity evaluation (DSET) has been proposed to simulate sub-query allocation and cost optimization for a distributed database query processing environment. Experiments have shown that dynamic evaluation of selectivity factor of sub query operation has significantly reduced the total query cost than its static estimation.

Keywords
Distributed database, query optimization, cardinality, selectivity factor, static Model, DSET etc.

1. INTRODUCTION
Distributed database systems design and query optimization has been an active area of Database Research for many decades [1]. In query processing in distributed database system, the query is decomposed into a group of sub-queries that are to be executed on different sites [2]. The aim of the query optimization is to decide a least cost query execution plan among various feasible plans. One major factor that affects the performance of an execution strategy is the size of an intermediate fragment produced, while executing a sub-query operation. After allocating these sub-operations on different sites, resultant relations are to be generated. Next we need to estimate the selectivity factor of these relations. Optimizer requires these estimates for choosing a least cost query operation allocation plan. On the basis of cost model, optimizer chooses the execution plan having the query cost close to the optimal [3].

2. RELATED WORK
Faiza and Yahya have proposed a statistical method for estimating the cardinality of the resulting relation obtained by relational operator by using sample based estimation that execute the query to be optimized on small samples of real database and use the results of these trials to determine cost estimates [4].

Areerat and Jarernsri have proposed Exhaustive Greedy (EG) algorithm to optimize intermediate result sizes of join queries. Most intermediate result sizes of join queries estimated by the EG algorithm are comparable to the results estimated by the Exhaustive Search algorithm (ESU)that is modified to update join graphs [5].

Fan and Mi Xifeng have designed a new algorithm based on heuristic optimization that can significantly reduce the amount of intermediate result data. The basic idea of this algorithm is based on relational algebra equivalence transformations to raise the connecting and merging operations in the query tree [6].

Gurvinder Singh et al. have proposed a stochastic model simulating a Distributed Database environment and shown benefits of using innovative Genetic Algorithms (GA) for optimizing the sequence of sub-query operations allocation over the Network Sites. Also, the effect of varying Genetic Parameters on Solution's quality is analyzed [7].

Rajinder singh et al. has highlighted a design of a probabilistic solution to the operation allocation problem of Distributed Databases. They highlight the design and implementation of one such model, Genetic Algorithm for sub query Allocation (GA_SA), which is a modest effort to stochastically simulate optimization of retrieval transactions for a distributed query [8].

Ridhi kapoor has described the selectivity and cost estimates in query optimization in distributed databases. They have discussed the various cost formulations to evaluate the cost of execution plans and then executing the plan with the minimum cost to the objective function [9].

Carlo et.al has proposed a method for estimating the size of relational query results. The approach is based on the estimates of the attribute distinct values. In particular, the capability of analytic method to estimate selectivity factors of relational operations is considered. They also presented some experimental results on real databases which show the promising performance of analytic approach [10].

3. DISTRIBUTED QUERY OPTIMIZATION
In distributed query optimization, one of the major components is generation of sub-query allocation plan. A complex distributed query needs to be divided into a number of smaller, simpler sub-queries. These sub-queries need to be executed on various different sites of distributed database, in order to minimize total cost of the query. The total cost that will be incurred in processing the query is a good measure of resource consumption. In a distributed database system, the total cost includes CPU, I/O and communication cost

that needs to be minimized. An optimizer's cost model includes cost functions to predict the cost of operators, statistics and base data and formulas. The cost is in the terms of execution time, so a cost function represents the execution time of a query [1]. A query optimizer generates a good query execution strategy that involves three phases. First is to find a search space which is a set of alternative execution plans for query. Second is to build a cost model that compares costs of different execution plans. Finally, it explores a search strategy to find the best possible execution plan among all using cost model [1].

Query optimization provides a quick way of answering queries for which the size of answer is of interest in its own right. The size of the intermediate relations that are produced during the execution is the main factor affecting the performance of a query execution strategy [5]. The size of the intermediate relations is based on the evaluation of selectivity factor of sub-operations.

4. SELECTIVITY ESTIMATION OF RELATIONAL OPERATIONS

Selectivity estimation is an integral part of query optimization. The selectivity factor of an operation is the number of tuples of an operand relation that participate in the result of that operation is denoted SFOP, where OP denotes the operation. The selection is usually based on the cost estimates of alternative plans, which in turn are based on the selectivity estimates of operators. Selectivity evaluation in turn depends on cardinality of fragments generated in the query. The selectivity estimation is based on statistical information about the base relations and formulas to estimate the cardinalities of the results of the relational operations [4]. There is a direct relationship between the precision of the statistics and the cost of managing them.

4.1 Selectivity formulations

The following formulae for relational operations were used to evaluate selectivity factor of various sub-query operations like selection, projection and join as per Ozsu's Model [1]. Here 'SF' and 'A' , represents selectivity factor and attribute respectively, 'card' represents cardinality of result and 'R' and 'S' represent two relations.

Table 1. Selectivity formulae

S.No	Operations	Formulae
1	Selection	$SFs = \frac{card(\sigma_A(R))}{Card(R)}$
2	Projection	$SF_P = \frac{card(\pi_A(R)}{card(R)}$
3	Join	$SF_J = \frac{card(R \bowtie_{A=B} S)}{max(card(R),card(S))}$

4.2 Database Statistics

The estimation of size of intermediate results of relational algebra is based on statistical information about the base relations and formulas to predict the cardinalities of the result of relational sub operations.

The size of each tuple of the relation is presumed to be 1KB.

No of base relations = 3

No of operations = 8

No of sites =3

I/O, CPU and communication coefficients are relative coefficients.

I/O speed coefficients = [1, 1.1, 1.2]

CPU speed coefficients = [1.1, 1, 1]

Communication speed coefficients =[0 10 12, 10 0 11, 12 10]

Size of each base relation =100 KB

4.3 Database example

Experimental database has taken from [1].

Table 2. EMPLOYEE Relation

EMP_NO	EMP_NAME	TITLE	COUNTRY
E1	Ling	Elec. Eng.	Toronto
E2	Smith	Syst. Anal.	New York
E3	Joe	Mech. Eng.	London
E4 . . E100	Davis Joe	Mech. Eng. Comp. Eng.	London . . London

Table3. ASG Relation

EMP_NO	PROJ_NO	RESP	DUR
E1	P1	Engineer	12
E2	P1	Analyst	24
E3	P3	Consultant	10
E3 . . E100	P4 P100	Engineer Engineer	18 . . 9

Table 4. PROJECT Relation

PROJ_NO	PROJ_NAME	BUDGET
P1	Instrumentation	15000
P2	Database developer	13000
P3	CAD/CAM	25000
P4 . . P100	Maintenance CAD/CAM	31000 . . 25000

4.4 Query:

π_{EMP_NAME} (($\pi_{EMP\_NO,\ EMP\_NAME}$

($\sigma_{COUNTRY=London}$ (EMPLOYEE))) $X_{EMP\_NO=EMP\_NO}$

($\pi_{EMP\_NOEMP\_NAME}$($\sigma_{RESP=Engineer}$ (ASG))))$X_{PROJ\_NO=PROJ\_NO}$

((π_{EMP_NO,EMP_NAME}($\sigma_{PROJ\_NAME=CAD/CAM}$(PROJECT))))

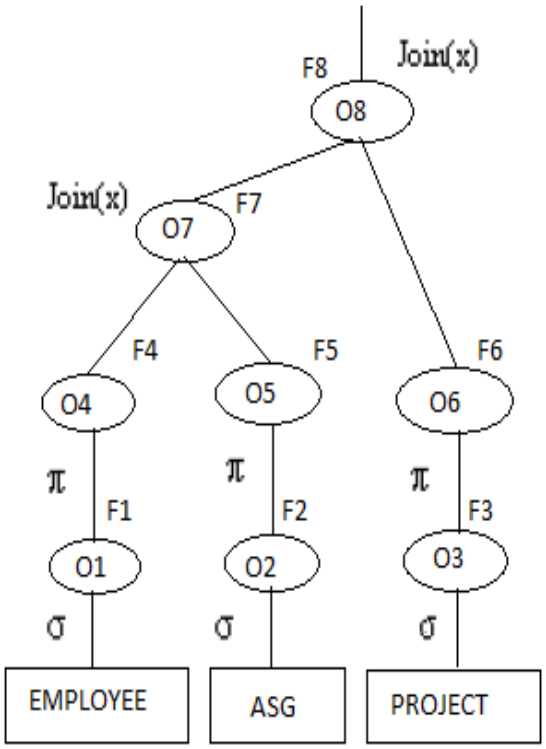
4.5 Operator Tree

The set of operations (sub-queries) generated in response to a query can be represented by an operator tree. Nodes of operator tree represent various operations and lines represent cost (based on size of fragment) of operation sequence. A site's Local CPU and I/O costs are proportional to the size (in bytes) of data processed and communication costs depend on communication coefficients between a pair of sites and bytes of blocks moved.

No of Operations: O1, O2…O8

No of Intermediate fragments: F1, F2….F8

Base relations: EMPLOYEE, ASG, PROJECT.



5. STATIC MODEL FOR SELECTIVITY ESTIMATION

Many of the query processing strategies in distributed databases are static in nature i.e., the strategy is completely determined on the basis of a priori estimates of the selectivity factor of sub query operations and it remains unchanged throughout its execution [11]. Due to this, the cardinality of intermediate fragments is large.

The pre-existing main simulator allocates sub operations to sites based on the database statistics assuming a set 'S' of data distribution sites, a set 'R' of relations/fragments stored on those sites[6]. In this simulator, the following array of selectivity factor of sub-operations of the query is statically fed to the simulator vide a input data file.

Selectivity factor of various sub-query operations = [0.7, 0.7, 0.7, 0.9, 0.9, 0.9, 0.35, 0.2].

For each operation the size of intermediate fragment is calculated by use of prefixed selectivity values for those operations [7].

Sub-Query Operation 1:

($\sigma_{\text{COUNTRY=London}}$ (EMPLOYEE))→F1, Tuples: 100 x 0.7(P_s) = 70

Sub-Query Operation 2:

($\sigma_{\text{RESP=Engineer}}$ (ASG)) →F2, Tuples: 100 x 0.7(P_s) = 70

Sub-Query Operation 3:

($\sigma_{\text{PROJ\_NAME=CAD/CAM}}$ (PROJECT))→F3,Tuples: 100x 0.7(P_s) = 70

Sub-Query Operation 4:

($\pi_{\text{EMP\_NO, EMP\_NAME}}$ (F1)) →F4, Tuples: 70 x 0.9(P_p) = 63

Sub-Query Operation 5:

($\pi_{\text{EMP\_NO, EMP\_NAME}}$ (F2)) →F5, Tuples: 70 x 0.9(P_p) = 63

Sub-Query Operation 6:

($\pi_{\text{EMP\_NO, EMP\_NAME}}$(F3)) →F6, Tuples: 70 x 0.9(P_p) = 63

Sub-Query Operation 7:

(f4 ⋮X⋮ f5) (EMP_NO=EMP_NO) → F7, Tuples: 63 x

0.35 (P_j) = 22

Sub-Query Operation8:

(f7 ⋮X⋮ f6) (PROJ_NO=PROJ_NO) →F8, Tuples: 63 x 0.2(P_j) = 13

6. DYNAMIC SELECTIVITY ESTIMATION TOOL (DSET)

DSET is a small simulator which feeds to the main simulator which allocates sub-queries to various sites. The major goal of the DSET is to evaluate selectivity factor of sub operations dynamically that can further helps in estimating the intermediate relation sizes of the similar kind of queries and thus can also reduce the response time of that queries. This simulator created three base relations, populated them with instance data and then took sub-query operations and using MATLAB-SQL interface to embed SQL code for selection, projection and join operations and estimated size from generated fragments. In case of DSET, cardinality is evaluated for intermediate results of the query by calculating selectivity factor at run time using selectivity formulae in table1. The overall cost of the query is directly proportional to the cardinality of the intermediate results. This cardinality is used in formulations mentioned above to calculate the selectivity factor of sub query operation more accurately.

Steps involved are:

First step: 3 base relations are created using SQL commands.

```
CREATE TABLE table_ name
(
column_name1 data _type (size),
column_name2 data _type (size),
....
);
```

Second step: Number of rows are inserted to the relations in order to calculate size of the base relations and to perform sub-operations to calculate cardinality of the resultant relations. Size of a relation = tuple size * number of tuples in a relation.

The basic difference from static model was that instead of feeding input data file giving intermediate fragment sizes, the operations are implemented in MATLAB/SQL code created intermediate relations and actually calculated sizes and hence the selectivity. Then this selectivity is dynamically fed to the operation allocator simulator.

6.1 Experimental data

After applying sub-operations (selection, projection and join) on EMPLOYEE, ASG AND PROJECT relations mentioned above, sizes of intermediate relations found to be:

Table 5: Size of intermediate relations

Relations	Size(KB)
EMPLOYEE	100
ASG	100
PROJECT	100
F1	58
F2	56
F3	53
F4	48
F5	45
F6	43
F7	12
F8	6

Sub-Query Operation 1:

Selectivity factor of selection operation on relation EMPLOYEE

$$SFs(EMPLOYEE) = \frac{card(F1)}{Card(EMPLOYEE)}$$

SFs = 58/100 =0.58

Sub-Query Operation 2:

Selectivity factor of selection operation on relation ASG

SFs (ASG) = 56/100 =0.56

Sub-Query Operation 3:

Selectivity factor of selection operation on relation PROJECTION

SFs (PROJECT) = 53/100 =0.53

Sub-Query Operation 4:

Selectivity factor of projection operation on fragment F1

$$SFP(F1) = \frac{card(\pi_A(F1))}{Card(F1)}$$

SFp = 48/58 = 0.82

Sub-Query Operation 5:

Selectivity factor of projection operation on fragment F2

SFp (F2) = 45/56= 0.8

Sub-Query Operation 6:

Selectivity factor of projection operation on fragment F3

SFp (F3) = 43/53= 0.81

Sub-Query Operation 7:

Selectivity factor of join operation on fragments F4 and F5 $SFJ(F4, F5) = \frac{card(F4 \bowtie_{EMP\_NO=EMP\_NO} F5)}{Max(card(F4), card(F5))}$

SF_J = 12/max (48, 45) = 0.24

Sub-Query Operation8:

Selectivity factor of join operation on fragments F7 and F6

SF_J = 6/max (12, 43) = 0.14

7. EXPERIMENTAL RESULTS

It highlights the fact that dynamic model for selectivity evaluation helps in reducing the overall cost of the query by dynamically calculating the cardinality of intermediate relations more accurately. Experimental results have shown that accuracy of dynamic evaluation of selectivity factor is comparable to the static estimation of selectivity factor.

Selectivity factor of selection operation – decrease by 20%

Selectivity factor of projection operation – decrease by 10%

Selectivity factor of join operation – decrease by 30%

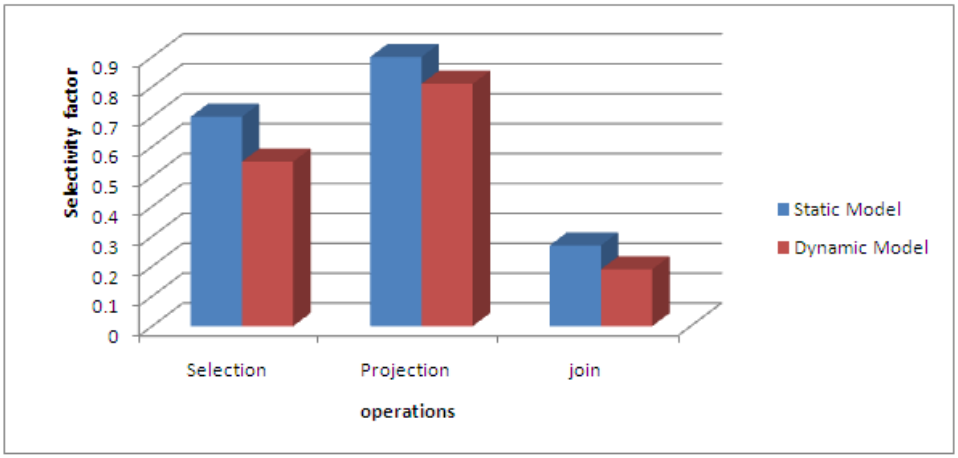


Fig1: Static vs. Dynamic model for selectivity evaluation

8. CONCLUSION

The aim of the experimental work was to analyze the effect of dynamic selectivity evaluation on the reduction of overall cost of the query. The advantage of using DSET was that size of intermediate relations calculated more accurately than static method. Hence, it resulted into lesser cost of sub-query. Finally, when the total cost of all sub-query operations on the various sites are added, the benefits achieved in the range of ten to thirty percent for various sub-operations selection, projection and join.

9. REFERENCES

[1] M.Tamer ozsu, Patric Valduriez ”Principles of Distributed Database Systems”, springer, 2010.

[2] B.M. Monjurul Alom, Frans Henskens and Michael Hannaford “Query Processing and Optimization in Distributed Database Systems”, IJCSNS, September 2009.

[3] Manik Sharma, Gurdev Singh, and Rajinder Virk. "Analysis of Joins and Semi Joins in a Distributed Database Query." International Journal of Computer Applications (IJCA), Vol. 49, Number 16, 2012.

[4] Faiza Najjar and Yahya slimani” Cardinality estimation of distributed join queries”2002.

[5] Areerat Trongratsameethong, Jarernsri L. Mitrpanont,” Exhaustive Greedy Algorithm for Optimizing Intermediate Result Sizes of JoinQueries”, IEEE, 2009.

[6] Fan Yuanyuan, Mi Xifeng”Distributed database System Query Optimization Algorithm Research”, IEEE, 2010.

[7] Rajinder Singh, Gurvinder Singh, Varinder Pannu virk” Optimized Access Strategies for a Distributed Database Design”, IJDE, 2011.

[8] Rajinder Singh, Gurvinder Singh, Varinder Pannu virk, ”A Stochastic Simulation of Optimized Access Strategies for a Distributed Database Design”, IJSER, November2011.

[9] Manik Sharma, Gurvinder Singh, Rajinder Singh, Gurdev Singh. 2013. "Stochastic Analysis of DSS Queries for a Distributed Database Design." International Journal of Computer Applications. Vol. 83 Issue 5.

[10] Carlo Dell’ Aquilla, Ezio Lefons, Filippo Tangorra,” Analytic-based Estimation of Query Result Sizes”, 2005.

[11] Peter Bodorik, J. Spruce Riordon, Member, IEEE, and James S. Pyra, “Deciding to Correct Distributed Query Processing”, IEEE, June1992.

[12] Abhijeet Raipurkar, G.R. Bamnote,” Query Processing In Distributed Database Through Data Distribution, International Journal of Advanced Research in Computer and Communication Engineering, Feb 2013.

[13] Manik Sharma, Gurdev Singh. 2012, “Analysis of Joins and Semi-joins in Centralized and Distributed Database Queries”. IEEE International Conference on Computing Sciences (ICCS), 2012.

[14] Sharma, Manik, Gurdev Singh, and Harsimran Kaur. "A Study Of BNP Parallel Task Scheduling Algorithms Metric's For Distributed Database System." International Journal of Distributed and Parallel Systems 3.1 (2012): 157.