IJCAT - International Journal of Computing and Technology
Volume 1, Issue 1, February 2014
www.IJCAT.org

Optimization of Large Join Query using Heuristic Greedy Algorithm

[1] Vishal P. Patel, [2] Hardik R. Kadiya

[1] PG Student of Computer Department, Merchant Engineering College, Basna, Gujarat,India

[2] Faculty of Computer Department, Merchant Engineering College, Basna, Gujarat, India

Abstract - Sql Statements are used to retrieve data from the database. One can get the same results by writing different sql queries. But use of the best query is important when performance is to be considered. So you need to use sql query tuning based on the requirement. Asqueries are stated in non-procedural manner, the need of optimizer arises that transform the straight forward translation of a query in to a cost-effective evaluation plan. Due to their high evaluation costs, joins are a primary target of query optimizers.Use of Heuristic to cut-down alternatives and Greedy Algorithm for finding suboptimal solution in less time rather than using dynamic programming and branch and bound to find optimal solution in more time than greedy algorithm.

***Keywords* - Query Optimization, Relational Databases Cost-Based optimization, Heuristic greedy algorithm, Hybrid query optimization.**

1. Introduction

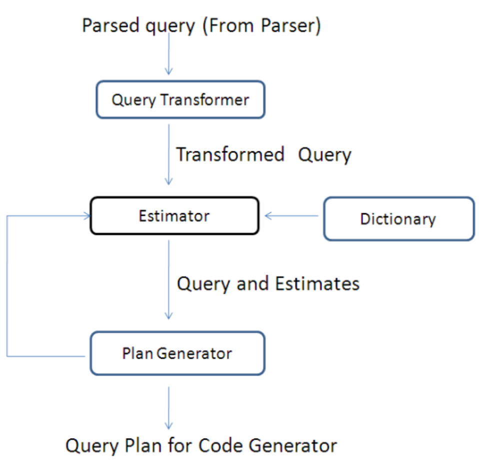


Fig. 1: Optimizer of a commercial RDBMS[1]

First the parsed query must pass the query transformer inside the optimizer the query transformer rewrite the query using heuristic like[2]

(1) Perform selection and projection as early as possible
(2) Predicate pushdown
(3) Subqueryunnesting

Example:SelectA.id,u from A where A.id in (select Aid from B where V=0); Can be transformed in to query using join method Select A.id,u from A inner join B on (A.id=B.id) where v=0;

Next the transformed query passes the estimator. This section estimates the costs (e.g. number of rows) of different operation which might be relevant to execute the query. Therefore a dictionary (which contains statistical information about the data) can be used to be able to estimate the number of rows which match a where clause.Finally, the plan generator is determining the expected optimal query plan to execute the query.

2. Database Statistics

Success of estimation depends on statistical information DBMS holds. Keeping statistics current can be problematic. If statistics updated every time tuple is changed, this would impact performance, so DBMS provide statistical information on base data.DBMS could update statistics on a periodic basis, for example nightly, or whenever the system is idle. This could lead to inaccurate estimates.Query optimization tries to find best possible plan within a minimum amount of time using mostly semi accurate statistical information.

3. Join Scheduling for Query Optimization

Algorithm applied to explore search space and determine the best query execution plan(QEP) based on join selectivity

Join selectivity=ratio of the number of tuples in the result/number of tuples in the Cartesian product.

Classes of strategies solve problem of join scheduling[4]

<u>Deterministic Strategy</u>: Starting from base relation, joining one or more relations at each step till complete plans are obtained.

Example: Dynamic Programming, Greedy Algorithm

Randomized Strategies: These strategies do not guarantee optimal plan but they avoid high cost optimization in terms of memory and time consumption Example: Iterative improvement, Simulated Annealing.

4. Explain Statement in SQL

An explain statement in SQL is a statement which consists of the keyword “explain” followed by a select statement. The select statement will then be parsed by the parser of the RDBMS. Afterwards, the optimizer will decidewhich one is the expected optimal query plan to execute the query.

Example: Explain select A.id ,B.id, u, v from A inner join B on (A.id=B.Aid)[5]

QUERY PLAN

Hash Join (cost=1.07. . . 2.18 rows=3 width=6)
Hash Cond :(b.aid=a.id)

→Seq scan on b (cost = 0.00...1.07 width=8)
Filter: (p=0)
→ Hash (cost=1.03…1.03 rows=3 width=6)
→Seq scan on a (cost=0.00..1.03 rows=3 width=6)

5. Shapes of join Query Tree

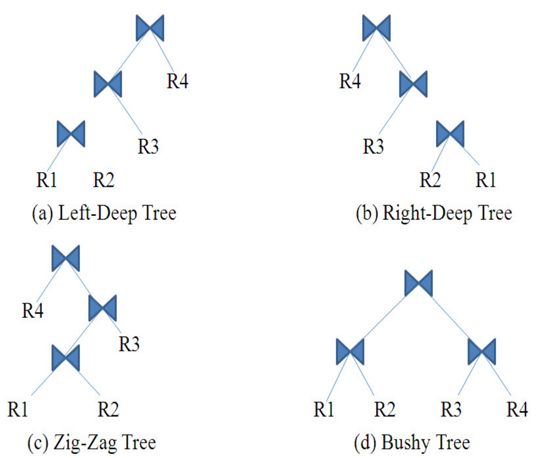


Fig. 2:Shapes of join Query Tree[6]

Number of possible join tree for a given n relation is determined by using (C_{n-1}*n!) where Cn is the Catalan number that is satisfied using following formula

$C_{n=}$ ((2n)!) / ((n+1)!*n!)[7]

Require to proceed further by using Left-deep tree is preferable because by using Left-deep tree we required to consider only(n!) possible tree where n is the number of relation in a query. By using Left-deep tree no require to store intermediate result and pipeline can be possible. In left-deep tree all leaf node as a relation and all intermediate node as a join operator.

Table 1. Number of Possible Trees [8]

Number of Relations(n)	Cn-1(Catalan Number)	Join Trees(Cn-1*n!)	Left-Deep Trees(n!)
2	1	2	2
3	2	12	6
4	5	120	24
5	14	1680	120
6	42	30240	720
7	132	665280	5040
8	429	17297280	40320
10	4862	17643225600	3628800

5.1 Transformation rules for Join operation[9]:

- **Join Method choice:** A $\bowtie_{\text{method i}}$ B → A $\bowtie_{\text{method j}}$ B
- **Join Commutative:** A ⋈ B → B ⋈ A
- **Join Associability:** (A ⋈ B) ⋈ C → A ⋈ (B ⋈ C)
- **Left Join Exchange:** (A ⋈ B) ⋈ C → (A ⋈ C) ⋈ B
- **Right Join Exchange:** : A ⋈ (B ⋈ C)→ B ⋈ (A ⋈ C)
- **Swap:** exchange the positions of two arbitrary positions in the sequence
  R1⋈ R2⋈ R3⋈ R4 ⋈R5 → R1⋈ **R4**⋈ R3⋈ **R2**⋈ R5
- **3-cycle:**cyclic rotation of three arbitrary position in the sequence.
  R1⋈ R2⋈ R3⋈ R4 ⋈R5 → **R5**⋈ R2⋈ **R1**⋈ R4⋈ **R3**

5.2 Shapes of Query[10]

1) Star Query: In star query there is a subset of attributes common to all relations in the query

2) Chain Query: In chain query all except two queries have common attribute with exactly two other relations but the first and last relations have attributes in common with only one other query. Each attribute is common to at most two relations.

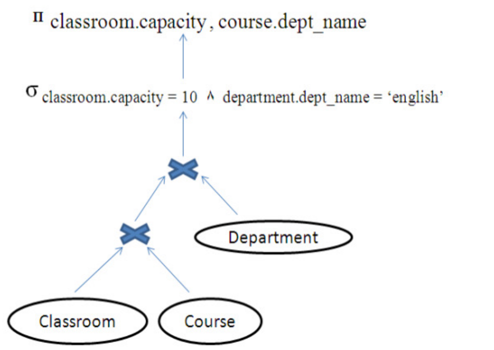
3) Circular Query: In a circular query, each query has common attribute with exactly two other relations and each attribute is common to at most two relations.

4) Clique query: In a clique Query, every pair of relation has a unique subset of common attributes. Proceed further by using chain query is preferable because most database operation of type chain query

6. Experiments & Results

6.1 Example (Cartesian Join):

Select classroom.capacity,course.dept_name,department.dept_name FROM classroom,course,department WHERE classroom.capacity=10 AND department.dept_name='english';



Initial Query Tree

Select T1.CAPACITY , T2.DEPT_NAME , T0.DEPT_NAME from (Select DEPT_NAME from DEPARTMENT where (department.dept_name = 'english')) as T0 , (Select CAPACITY,BUILDING,ROOM_NUMBER from CLASSROOM where (classroom.capacity = 10)) as T1 , (Select DEPT_NAME,COURSE_ID from COURSE) as T2

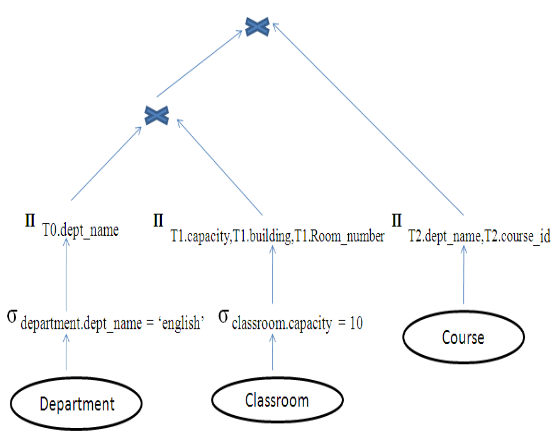


Fig.3: Initial Query Tree &Using Algorithm

Table 2. Database Statistics (Cartesian Join Example)

Table Name	No of Rows	Avg. Row Length	Table Size	Affected Rows	Result Size
Department (T0)	20	30	600	1	30
Classroom (T1)	30	20	600	5	100
Course (T2)	200	37	7400	200	7400

Cost of Initial Query Tree [Space usage]=
A [Cost of crossjoin T1and T2] + B [Cost of cross join A and T0] +C [Number of rows after apply selection on B]
=A [{30 * 200} * {20+37}] + B [{6000 * 20}* {57+30}] + C[{1000*87}]
=A[342000] + B [10440000] + C [87000]

Cost of Query Tree by Algorithm[Space usage] =
A[Cost of applying selection on T0] + B[Cost of applying selection on T1] + C[cost of apply cross join on A and B]+D[cost of apply cross join on C and T2]
=A[{1*30}]+B[{5*20}]+C[{1*5}*{30+20}]+D[{5*20} *{50+37}]
=A[30]+B[100]+C[250]+D[87000]

6.2 Example (Equi Join):

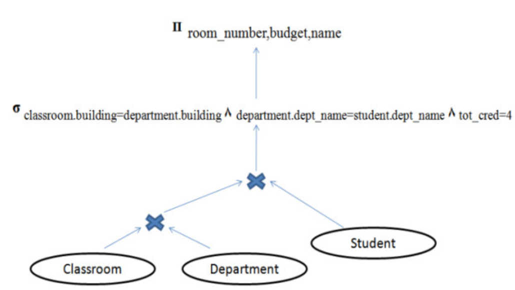
Table 3. Database Statistics (Equi Join Example)

Table Name	No of Rows	Avg Row Length	Table Size	Affected Rows	Result Size
Student(T0)	2000	29	58000	12	348
Classroom(T1)	30	20	600	30	600
Department(T2)	20	30	600	20	600

Cost of Initial Query Tree [Space usage] =
A [Cost of cross join T1 and T2] + B [Cost of cross join A and T0] + C[Number of rows after apply selection on B]
=A[{30*20}*{20+30}]+B[{50*2000}*{50+29}]+C[{10*79}]
=A[30000]+B[7900000]+C[790]

Cost of Query Tree by Algorithm [Space usage] =
A [Cost of applying selection on T0] + B [Cost of applying join on T2 and A] +
C [Cost of applying join on B and T1]
= A [{12*29}] + B [12*(29+30)] + C [10*(29+20+30)]
= A [348] +B [708] +C [790]

SELECT room_number, budget, name FROM classroom, department, student WHERE classroom.building = department.building AND department.dept_name = student.dept_name AND tot_cred = 4



Initial Query Tree

SELECT ROOM_NUMBER , BUDGET , NAME from (Select NAME,ID from STUDENT where (tot_cred = 4)) as T0 , (Select ROOM_NUMBER,BUILDING from CLASSROOM) as T1 , (Select BUDGET,DEPT_NAME from DEPARTMENT) as T2 where (T1.BUILDING = T2.BUILDING) and (T2.DEPT_NAME = T0.DEPT_NAME)

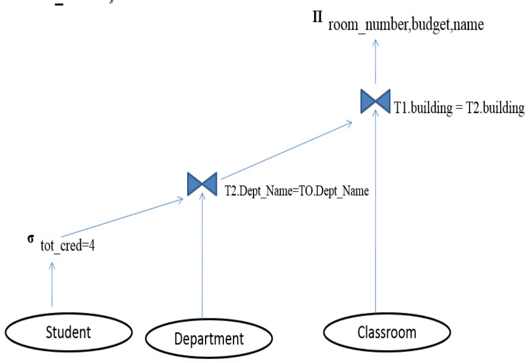


Fig.4: Initial Query Tree & Using Algorithm

7. Conclusions and Future Work

Reduce intermediate result size ultimately reduces the execution time. Experiment result shows Combine Heuristic and Greedy approach provide better performance for optimization of Large join query.Optimization only on select-project-join queries also need to handle complex queries (e.g. Top-k query, Group by, aggregations, materialized view, recursive query, dynamic query).Optimize query on centralized system can be extended to implement on parallel database and Distributed Database.

References

[1] Dr.G. R. Bamnote, Prof. S. S. Agrawal, “Introduction to Query Processing and Optimization”IJARCSSE Volume 3, Issue 7, July 2013 ISSN: 2277 128X

[2] Alon Y. Levy, Inderpal Singh Mumick, YehoshuaSagiv, “Query Optimization by Predicate Move-Around”, AT&T Bell Laboratories, Proceedings of the 20th VLDB Conference, Santiago, Chile, 1994.

[3] Nicolas Bruno, SurajitChaudhuri,“Efficient Creation of Statistics over Query Expressions” Data Engineering, 2003. IEEE 19th International Conference, On Page(s): 201-212 ISBN:0-7803-7665-X.

[4] Prof.M.A.Pund, S.R.Jadhao, P.D.Thakare, “A Role of Query Optimization in RelationalDatabase”, International Journal of Scientific & Engineering Research, Volume 2, Issue 1, January-2011 1ISSN 2229-5518.

[5] Thomas Mayer, “Analyzing Plan Diagrams of Database Query Optimizers”, KIT, Institute for Programmstrukturen und Datenorganisation (IPD),D-76131 Karlsruhe, Germany.

[6] AlaaAljanaby, EmadAbuelrub, and Mohammed Odeh, “A Survey of Distributed Query Optimization” The International Arab Journal of Information Technology, Vol. 2, No. 1, January 2005.

[7] “cost based plan selection enumerate estimate select”, "Database Systems: The Complete Book (second edition)" by H. Garcia-Molina, J. D. Ullman, and J. Widom (ISBN-13: 978-0131354289)

[8] SurajitChaudhuri, “An Overview of Query Optimization in Relational Systems”, seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems, Pages 34-43, ISBN:0-89791-996-3

[9] Yannis E. Ioannidis,Younkyung Cha Kang, “Left-Deep Vs Bushy Trees: ACM 0-89791-425-2/91/0005/0168 ACM

[10] MostafaPilehvar, “A new approach to join reordering in query optimization” Concordiauniversity, Montreal, Canada, March 2005

[11] N. Satyanarayana, Sk.Sharfuddin, Sk.JanBhasha, “New Dynamic Query Optimization Technique InRelational Database Management Systems” International Journal Of Communication Network Security, Issn: 2231 – 1882, Volume-2, Issue-2, 2013

[12] Prof. Miss. S. D. Pandao, Prof. A. D.Isalkar, “Multi Query Optimization Using Heuristic Approach Heuristic Approach” International Journal of Computer Science and Network (Ijcsn) Volume 1, Issue 4, August 2012 Www.Ijcsn.Org Issn 2277-5420

[13] JyotiMor,InduKashyap, R. K. Rathy, “Analysis of Query Optimization Techniques in Databases”International Journal of Computer Applications (0975 – 888)Volume 47– No.15, June 2012

[14] CJ.Date, “An Introduction to database systems”Adission wisely

[15] Abraham Silberschataz, Henry F. Korth& S. Sudarshan, “Databse system concepts” Mc Graw Hill

Mr. Vishal P. Patel, He graduated from L.C institute of technology Bhandu (Affiliated to H.N.G.U Patan), presently pursuing M.Tech. in computer engineering at MEC,Basna affiliated to Gujarat technological university, Gujarat, India. His interested area in object-oriented systems, database and computer algorithms

Mr. Hardik R. Kadiya,He is Asst.Professor in Merchant engineering college,Basna,Gujarat,India.He received bachelor degree in information technology from government engineering college,Modasa,Gujarat,India. And M.E from merchant engineering college,affiliated to Gujarat technological universityand his area of interest are image processing,database and computer network.