An algorithm of fragmentation optimization in distributed database

Igor Zhukov[1], Ivan Kravets[2]

[1] Professor, director of the institute of computer technologies, National Aviation University, Kosmonavta Komarova ave. 1, Kiev, 03058, UKRAINE, E-mail: zhukov@nau.edu.ua

[2] PhD student of department "Computer systems and networks", National Aviation University, Kosmonavta Komarova ave. 1, Kiev, 03058, UKRAINE, E-mail: me@ikravets.com

Developed a method of fragmenting large-size data based on analysis of requests to the database management system; algorithm of search of the shortest path for given vertices in multigraph. Proven that this algorithm allows to distribute data between databases more efficiently, and also to reduce time of executing requests.

Keywords – distributed database, fragmentation algorithm, orgraph, multigraph, weighted vertices.

I. Introduction

Recently more and more IT specialists are running into issues related to slow work of their databases. Some are resorting to hardware upgrades of the server which hosts database management system, and some are building huge cluster systems hoping that load balancing will solve all the problems.

But question about fragmentation becomes more interesting in cases when majority of RDBMS requests are (to some extent) remain the same while DB size grows exponentially. It might have practical applications for BDs in medical care, educational institutions, libraries and other organizations, where 90% of the data is of archive nature [1].

Goal of this article is to describe of the new algorithm of optimization of data fragmentation in distributed database (DDB).

II. Algorithm of optimization of data fragmentation

As of now, majority of specialists are designing BD and creating requests based on the data of small size, or fragmenting BD into different parts and don't think that ratios of data grows can be different from initial prognosis. With time, it can affect time of request execution in RDBMS.

Time of request execution can be written as Eq. (1)

$$\tau = \sum_{i=1}^{n} (\tau_{o_i} + \tau_{c_i} + \frac{V_i + \tau_b}{V_b}) + \tau_{con}, \qquad (1)$$

where n – number of tables in request; $\tau_{o,i}$ – time to open i-th table; $\tau_{c,i}$ – time to close i-th table; τ_b – time to read data block; τ_{con} – total time to connect; V_i – volume of i-th table; V_b – volume of the block.

To find the shortest path of joining tables in DDB let us consider SQL-request in the form of orgraph G:=(V,A), where V is a set of vertices which represents DB tables, and A – set of pairs of different vertices (edges) based on joining conditions of two tables.

As the next step it is necessary to build connected multigraph. For every it's vertice we will assign weight c_i – time to access and reading the table, every edge we will assign weight d_i – time to joining related tables (including total time to connect to different DDB). This way, to choose optimal path it is necessary to perform optimization for graph with weighted vertices and edges.

Task of graph optimization consists of choosing of the least weighted sub-graph under condition that resulting sub-graph is connected one:

$$f(G) = [\sum_{i=1}^{n} c_i + \sum_{j=1}^{m} d_i] * x_i \rightarrow \min, \qquad (2)$$

where $c_i = \tau_{o_i} + \tau_{c_i} + \frac{V_i + \tau_b}{V_b}$, c_i – weight of i-th vertice; n – number of vertices; m – number of edges; d_i – weight of j-th edge; x_i=1, if i-th vertice (table) can be fragmented, and 0 otherwise.

Based on resulting minimal path it is necessary to perform fragmentation of appropriate tables.

III. Example of using algorithm of optimization of data fragmentation

For analysis it was taken a request which joins five tables from three RDBs:

```
SELECT STRAIGHT_JOIN
    w.worker_firstname,
     w.worker_lastname,
    wd.workday_date,
    wju.jobauit_start,
    wju.jobauit_end,
     jc.jobclass_name,
    wdwr.workday_wagerate
FROM
    (workers w, workdays wd)
    LEFT OUTER JOIN worker_jobaudit wju ON
(wju.worker_id = w.worker_id AND wju.jobdate =
wd.workday_date)
    LEFT OUTER JOIN jobclasses jc ON (jc.jobclass_id
= wju.jobclass_id)
    LEFT OUTER JOIN workday_wagerates wdwr ON
(wdwr.workday_id = wd.workday_id)
WHERE
    w.worker_spared = 0 AND wd.workdate >
DATE_SUB(CURDATE(),INTERVAL 31 DAY)
ORDER BY
    wd.workday_date.
```

From SQL-request we see that request was written as if all the data are located within the same DB. It is achieved due to transparent fragmentation and properties of RDB [2]. Presenting this request as graph [3], and after performing conversion from tables into vertices, orgraph shown on Fig. 1 was obtained.

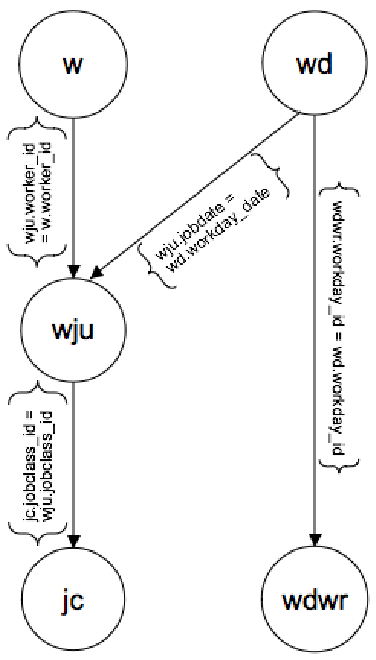
w
wd
wju
jc
wdwr
wju.worker_id = w.worker_id
wju.jobdate = wd.workday_date
wdwr.workday_id = wd.workday_id
jc.jobclass_id = wju.jobclass_id

Fig. 1 Orgraph based from request to RDBMS

Then connected multigraph with weighted vertices and edges was built (Fig. 2). It is an abstract representation of orgraph on Fig. 1 based on 3 RBDs. It schematically shows:

- DB_1-DB_3 – is distributed database for N=3 servers;
- c_1-c_{15} – weight for vertices;
- d_1-d_{35} – weight for edges;
- c_1-c_7-c_{13}-c_9-c_{15} – initial path with previous state of fragmented data;
- c_1-c_7-c_8-c_9-c_{10} – optimized path.

Calculated values of weights on vertices c_i based on Eq. 1 are shown in Table I. And weights on edges d_j include into them: time to connect related tables, delay between servers at network level and time of RDB authorization (Table II).

TABLE 1

WEIGHT FOR VERTICES

i	1	2	3	4	5	6	7	8
c_i, msec	5	-	2	-	1	-	32	13
i	9	10	11	12	13	14	15	
c_i, *msec*	11	14	-	-	2	-	2	

TABLE 2

WEIGHT FOR EDGES

j	1	2	3	4	5	6	7	8	9
d_i, *msec*	-	-	-	-	-	7	4	3	-
j	10	11	12	13	14	15	16	17	18
d_i, *msec*	-	-	-	-	196	-	-	-	158
j	19	20	21	22	23	24	25	26	27
d_i, *msec*	-	-	-	-	107	111	-	-	114
j	28	29	30	31	32	33	34	35	
d_i, *msec*	131	-	-	-	-	149	-	-	

It should be mentioned, that when fragmeting DB, it is not always possible to perform manipulation with all tuples from tables for the benefit of one or another database server. But this case is possible when fragmentation is done for the first time.

Then tuples of data from tables will be fragmented in such way, to put orgraph (Fig. 1) into the limits of the same BD. It will allow to avoid joining tables from different physical servers, which will indirectly reduce time of request execution.

The same way in this case: after analysing statistical data, which were gathered based on “evolutional algorithm” [4-13], it was allowed to fragment only the following tables: *worker_jobaudit* (c_3, $c_{8,}$ c_{13}) and *workday_wagerates* (c_5, $c_{10,}$ c_{15}).

Therefore, in Tables I and II weights on vertices and edges was set to "-", because paths which lead through them are impossible.

The final list of alternative paths is shown in Table 3, their number is nine and all of the traverse exactly five vertices, because request (Fig. 1) consists of joining of 5 tables.

TABLE 3

ALTERNATIVE PATHS

#	Path	Time, msec
1	1,7,3,9,5	530
2	1,7,3,9,10	432
3	1,7,3,9,15	548
4	1,7,8,9,5	383
5	**1,7,8,9,10**	**285**
6	1,7,8,9,15	401
7	1,7,13,9,5	617
8	1,7,13,9,10	519
9	1,7,13,9,15	635

The path #5 is optimized and have a minimal time for query execution. The initial path #9 – have 625msec. This is in more then 2 times.

Conclusion

Developed a method of fragmenting large-size data based on analysis of requests to the database management system; algorithm of search of the shortest path for given vertices in multigraph.

Proven that this algorithm allows to distribute data between databases more efficiently, and also to reduce time of executing requests.

Thereafter, this method Eq. (2) can be extended by a detailed study of indicators of load edges. It can be taken into account: the amount of data transferred between the two RDB of network protocols, performance of physical servers.

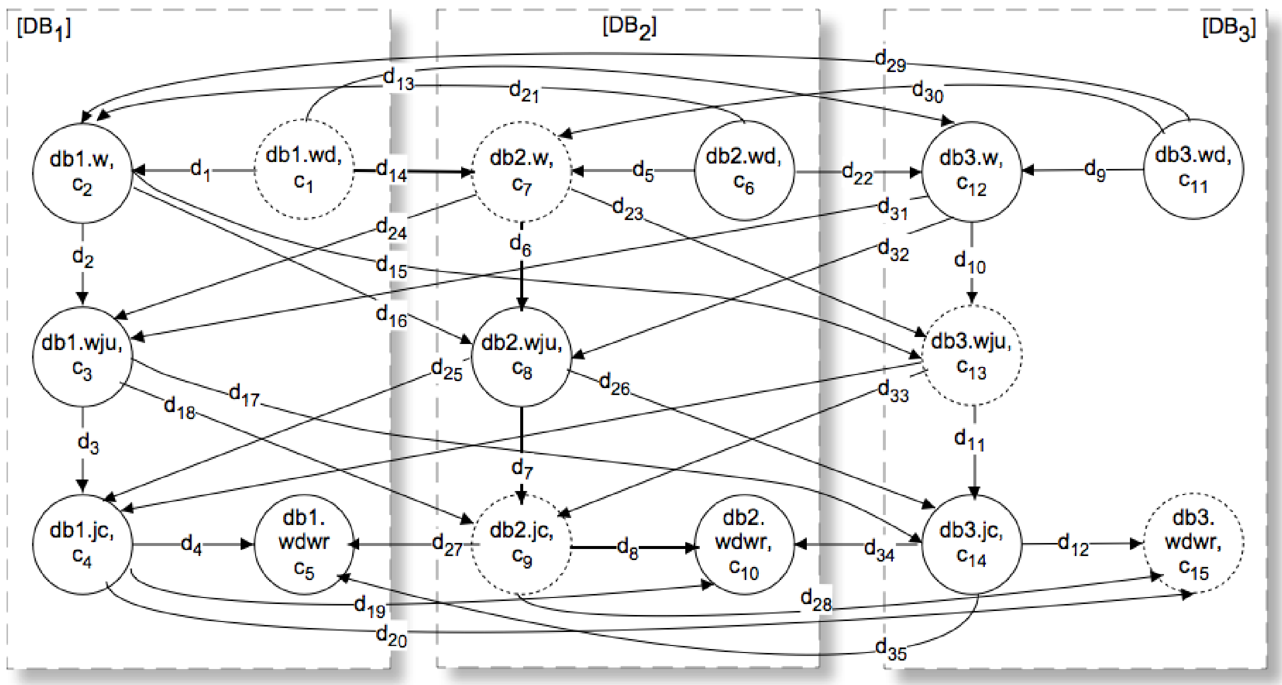

Fig. 2 Multigraph with weight for vertices and edges

References

[1] I.A. Zhukov, I.M. Kravets, “Distribution load database in the information-analytical system”, *The Problems of information and* , vol. 4(22),Kyiv,NAU,pp.56-61, 2007.

[2] I.A. Zhukov, I.M. Kravets, “Organization of distribution load database in the analysis and information system”, *International scientific technical conference “DESSERT-2009”, Radioelectronic and computer system*, vol. 5(39),Kharkiv,KhAI,pp. 25-30, 2009.

[3] M. N. Kyrsanov. “Graphs in Maple. Tasks, algorithms, programs”, Fizmatlit, Moscov, pp 29-39, 2007.

[4] Bäck, T., Fogel, D.B., Michalewicz, Z. (Editors), “Handbook of Evolutionary Computation”, *Institute of Physics Publishing*, Bristol and Oxford University Press, New York, pp. 113-132, 2006.

[5] Bäck, T., “Optimal mutation rates in genetic search”, *Proceedings of the 5th International Conference On Genetic Algorithms*, Ed. S. Forrest, Morgan Kaufmann, San Mateo, CA, pp. 2-8, 2001.

[6] Dumitrescu, D., Lazzerini, B., Jain, L.C, Dumitrescu, A., “Evolutionary Computation”, *CRC Press*, Boca Raton, FL. pp. 47-52. 2000.

[7] Goldberg, D.E., Deb, K., “A comparative analysis of selection schemes used in genetic algorithms”, *Foundations of Genetic Algorithms G.J.E. Rawlins (Ed.)*, Morgan Kaufmann, San Mateo, CA, pp. 69-93, 2001.

[8] Moldovan, G., “Reorganization of a Distributed Database”, *Babes-Bolyai University, Seminar of Models, Structures and Information Processing*, Preprint nr. 5, pp. 3-10, 2007.

[9] Mitchell, M., “An Introduction to Genetic Algorithms ”, *MIT Press*, Cambridge, MA, 1996.

[10] Oszu, M. T., Valduriez, P., “Principles of Distributed Database Systems”, *Prentice Hall*, Englewood Cli.s, NJ, p. 13, 2005.

[11] Makinouchi A., Tezuka M., Kitakami H., Adachi S. “The optimization Strategy for Query Evaluation in RDB/V1” , *Proc. 7th Int. Conf. Very Large Data Bases*, Cannes, France, pp. 518-529, Sept. 3-11, 2004.

[12] Piattini, M. and Diaz, O., “Advanced Database Technology and Design”, *Artech House*, Inc. 685 Canton Street Norwood, MA 02062, 2000.

[13] Weiss, G., “Multiagent System, A Modern Approach to Distributed Artifical Intelligence”, *MIT Press* , USA, 2000.