Business Process Optimization in Madrid City Council

JUAN M. GARITAGOITIA
Informática Ayuntamiento de Madrid (IAM)
Ayuntamiento de Madrid
Albarracín 33, 28037, Madrid
SPAIN
garitagoitiajm@madrid.es

NURIA GÓMEZ, CARLOS CUVILLO, LUIS F. MINGO
Universidad Politécnica de Madrid
Escuela Universitaria de Informática
Crta. de Valencia km. 7, 28031 Madrid
SPAIN
{ngomez,ccuvillo,lfmingo}@eui.upm.es

Abstract: While designing systems and products requires a deep understanding of influences that achieve desirable performance, the need for an efficient and systematic decision-making approach drives the need for optimization strategies. This paper provides the motivation for this topic as well as a description of applications in Computing Center of Madrid city Council. Optimization applications can be found in almost all areas of engineering. Typical problems in process, working with a database, arise in query design, entity model design and concurrent processes. This paper proposes a solution to optimize a night process dealing with millions of records with an overall performance of about eight times in computation time.

Key–Words: Process Optimization, DataBase Optimization, Information Retrieval, Business Process.

1 Introduction

Business Process Improvement (BPI) is a systematic approach to help an organization optimize its underlying processes to achieve more efficient results [4]. The methodology was first documented in H. James Harringtons 1991 book Business Process Improvement [1]. It is the methodology that both Process Redesign and Business Process Reengineering are based upon. BPI has been responsible for reducing cost and cycle time by as much as 90% while improving quality by over 60%.

In this paper we focus on process operators. The process operator is responsible to learn and perform the processes (work) necessary to achieve the objectives of the business plans that are created by Business Leaders. The responsibilities of the process operator follow the PDCA (plan, do, check, and act) cycle.

- Plan: The process operators - in collaboration with their Operational Manager, create and own their performance objectives. Process Operators are responsible to understand the performance objectives of the process they are to perform and the specifications of the product they are to produce.
- Do: Process operators are responsible to learn the processes (work) that they are to perform. They ensure the processes are performed to meet the process performance objectives and produce product that meets specification. As the Process Operators perform the processes, they are responsible to communicate to their Operational Manager (supervisor) the bridges that need to be built and the barriers that need to be removed to allow the process and Process Operator performance objectives to be met. Process and Process Operator performance metric data [2] is produced and collected as the process is performed.
- Check: The process operator periodically reviews the Key performance indicators (KPIs). The Process Operator makes adjustments to their work based on their actual performance compared to KPI targets. The Process Operator is responsible for identifying and reporting any performance issues and stopping production if necessary [3].
- Act: Process operators practice kaizen to continually challenge the process and communicate improvement suggestions to their operational manager (supervisor).

Database management systems have become a standard tool to hide implementation details for computer users of secondary storage management. They are designed to improve the productivity of application programmers and to facilitate data access by computer-naive end users. Exact optimization of query evaluation procedures is in general computationally intractable and is hampered further by the lack of statistical information about the database. Query evaluation algorithms must rely on heuristics. In this paper we state objectives of query optimization and present a procedure designed to structure the solution process applied to Madrid City Council.

2 Problem Description

Madrid city Council IT Service (IAM), an independent administrative organization, is in charge of managing, improving and carrying out the different implementation systems at Madrid city Council. In order to do this, all types of developing technologies and methodologies are used. Among the various applications developed by the personnel we can find the administrative proceedings for the citizens in Madrid. The completion period is the sticking point to some of the applications, due to the high volume of data managed. Many of the applications are exclusive in terms of execution time, so they cannot be launched in parallel. If we managed to reduce the execution time in some of them, there would be enough free space in our UCP, allowing us to benefit from the machine at a higher level. A process to validate accounting details, using the application that manages the income in Madrid city Council, needs to be executed on a daily basis. Running the income data, validating it and identifying possible mismatches are the main tasks of this process [6, 7].

The process is very easy in itself; it reads and gathers the data from various charts and obtains a list of possible errors. This list is verified by the Quality Control department, who refers to the corresponding department if a mismatch is found, solving the problem this way. A Charge is structured as follows: every debt (charge) is found inside what we call Cost. A cost contains 1 to N debts, from which nothing, part, or all of it, has been collected. At the same time, every debt can be broken down into several parts, as it can be paid in different times. Details of the various charges made by the administration arrive every day; these details go through the processes in charge of updating the database [5] and modifying de debts linked to the charges. In order to guarantee that the upgrading system is correct, the process under study is executed on a daily basis. It is at this point that we face the problem, as the process takes approximately between 160 and 200 minutes running time. We need to bear in mind that the process can only be executed once the rest have finished, as the ones executed earlier upgrade the database and cannot be launched simultaneously, for they would block this read only process. The process under study needs to be executed on a daily basis; what used to happen is that the earlier process took longer than usual due to the high volume of data, and there was practically no way to end it on time. Be advised that all processes need to be done by 8am as this is the time users start work. At this point, the process planning department faces the need to improve and reduce the process timing.

The process is implemented in Natural and executed in an IBM machine. As a database manager, we use DB2. The process is planned every day and executed in Batch, using JCL for this purpose. Its execution is monitored using terminals that are placed at the Data Processing Centre (CPD).

The software/tools environment is based on:

- Database IBM DB2 version 7
- Metric queries based on LISTSQL (NATURAL TOOLS FOR SQL)
- Process scheduler using Control-M from BMC-Software
- Operating system IBM/ZOS version 1.2

3 Experimental Benchmark

In the previous section we talked about how a charge is structured; we explained that every debt (charge) is found inside what we call cost. A cost contains 1 to N debts, and at the same time every debt is broken down into N elements. Having said this, we have an Entity Relationship Model composed by 3 tables, see figure 1.

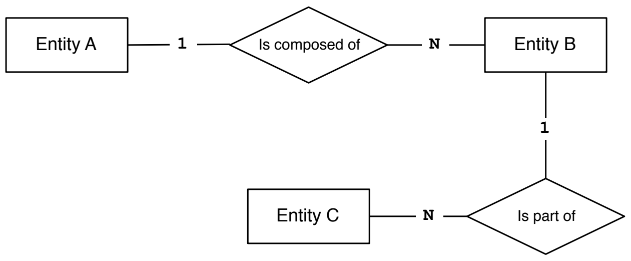


Figure 1: ER Model representing charges, debts and elements.

To facilitate the search of debts and extracts (parts of the debt), a view in DB2 was generated containing data from both charts. By doing this, we could have the data for all the debts and their extracts in the same view, see figure 2.

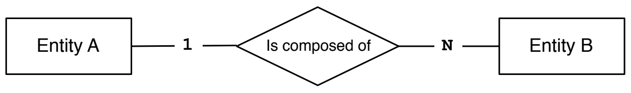


Figure 2: DB2 view representing debts and items.

A Relationship Entity Model made up of two entities with a relationship of $1 \cdots N$. To enable search optimization, the main key in entity A is passed to entity B, turning it into an index of this one. This model is currently duplicated, as we have part of the data in life charts and part in historical charts, both exactly equal. The total volume of information from the entities (life + historical) is approximately as follow:

- Entity A: 2.000 items
- Entity B: 10.000.000 items

The process in itself acts as follows: it reads each element in A in ascending order from the main code and accesses the elements in B to make the total calculation. Remember that we access entity B through the code inherited from A, which is also an index in B. Due to the Cartesian product, no join or inner join will be executed, as they could slow down the process.

4 Testing Results

Following the model of data explained in earlier sections and using measurement tools, we will see how much it costs to access the data, so we use tools that help us measure the cost of access. In both models the first priority is to access the data in entity A which will need to be scanned in only one command. The cost will be low thanks to the small size of the chart.

The cost of access is common in both diagrams, but differs substantially from here on. Here are the different models available to us.

4.1 Model 1

Using this model we access three charts: A, B and C. The cost of accessing entity A is calculated using figure 2 and 3. The cost of accessing entities B and C are calculated as follows:

- Cost of accessing entity B. For each element in entity A, we search all elements that have relation with elements in B, see figure 4.
- Cost of accessing entity C. Starting from B, we search all the elements in entity C that are related to each element in entity B, see figure 5.

4.2 Model 2

Using this model we access two charts, A and B (being this a join of charts B and C from model 1). The cost of accessing entity A is solved as we indicated earlier.

The cost of accessing entity B will be as follows (we need to observe that we search all elements in B that match up with each element in A). In the figure 6 we see that the cost of access is excessive.

5 Performance Execution Time

Using this structure as the starting point, the times of execution in the machine are in table 1.

If we look at these executions, we will find that the day it took the least was the 8th of May, 2012 with 145.40 minutes, and the day it took the longest was the 21st of April, 2012 with 213.18 minutes. This is a long time, if we take into account that this process cannot concur with any other, as it would produce a DeadLock.

6 Final Implementation

Due to the high amount of time involved in this process, we had to bring about a solution to reduce the time of execution. We realized that the process used a small group of attributes in entities A and B, specifically 1 attribute in A (which is also found in B) and 5 attributes in B. The process did organized counts with the attribute in A and it obtained a result. What we did was to create two views with the basic fields, one containing live data (with approximately 6 million elements) and one with the historical data (with approximately 4 million elements). Once the views for the entities are created, we make a call to the database manager (via JCL) to produce a massive unload of data (UNLOAD) to flat documents (adequately dimensioned), using the same format as we have in the database. As they are independent views, they can run in parallel.

To check the performance we measure the times.

- Unload of the first view, with a volume of data of approximately 4,448,590 elements (*)
- Unload of the second view (historical), with a volume of data of approximately 6,059,739 (*)

(*) The volume of data is more or less the same for each execution, with very little variations from one another:

We realize that the time for extracting data is considerably low. By running both unloads in parallel, the time counting the highest corresponds to the one that takes longer. If we look at the previous figure, we can see that the one that takes longer is the one dated 10th of May, 2012 with a timing of 20.18 minutes.

Once the data is downloaded in folders, we will only need to put them together in only one folder, arrange them by field and perform calculations. The volume of data in the folder is significant (about 10 million elements), but as we already have them inside the memory, there is no need to access the database to obtain data and calculate. In order to check its effectiveness, we again perform calculations of time, obtaining the following results.

Performance execution runtimes are obtained taking into account tables 2, 3, 4 and 5.

- the day the process took the longest was the 9th of May, 2012 with a total of 7 minutes.

- the day it took the longest was the 10th of May, 2012 with a total cost in time of 20.18 (unload) +5.15 (process) = 25.33 minutes.

- and the least was the 12th of May, 2012: 12.36 (unload) +5.25 (process) = 17.61 minutes.

If we compare this with the times of execution of the previous process, the decrease is significant, where the lowest and highest times of execution are:

Empirically, we can observe a decrease by 8 times execution time.

7 Conclusions

This paper proposes a solution to optimize a night process dealing with millions of records with an overall performance of about eight times in computation time. The process is very easy in itself; it reads and gathers the data from various charts and obtains a list of possible errors. This list is verified by the Quality Control department, who refers to the corresponding department if a mismatch is found, solving the problem this way.

Acknowledgements: We would like to thank Pedro Vizcaíno Moreno, Project Manager of Exploit Department at IAM, for his support in the proposed solution, for his help in data retrieval and for his great knowledge in the use of tools to obtain results. We also thank the Database Department at IAM for their ideas in the developed solution for this work. The process under study needs to be executed on a daily basis; what used to happen is that the earlier process took longer than usual due to the high volume of data, and there was practically no way to end it on time. Be advised that all processes need to be done by 8am as this is the time users start work. At this point, the process planning department faces the need to improve and reduce the process timing.

Two models have been studied to finally accomplish an eight times performance in computational time.

References:

[1] Harrington H. J., Business Process Improvement: The Breakthrough Strategy for Total Quality, Productivity, and Competitiveness, McGraw-Hill, ISBN: 0-07-026768-5, 1991.

[2] Hou Hong, You GuiFang, Song QinBao and Hao KeGang, The Research of Metrics Repository for Business Process Metrics, ISCID '09. Second International Symposium on Computational Intelligence and Design, vol.2, no., pp.168-171, 2009

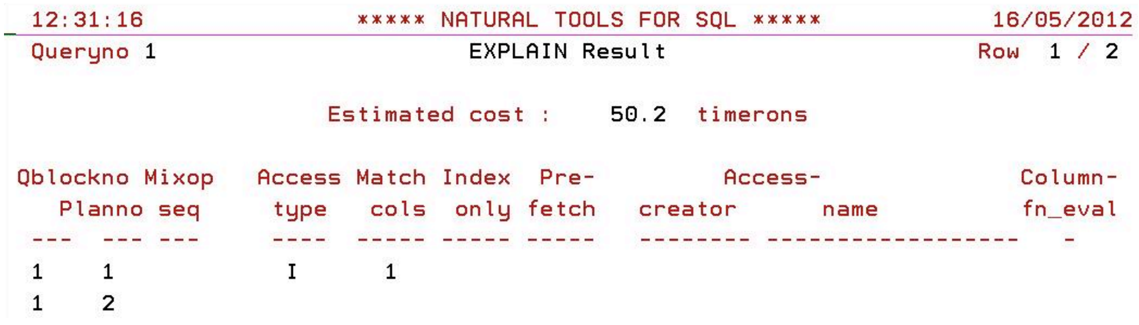
[3] Becker J., Bergener P., Delfmann P. and WeiB B., Modeling and Checking Business Process Compliance Rules in the Financial Sector, Thirty Second International Conference on Information Systems, pp:1-19, 2011. Proceedings of the 6th WSEAS Int. Conf. on Artificial Intelligence, Knowledge Engineering and Data Bases, Corfu Island, Greece, February 16-19, 2007 147

[4] Vergidis K., Tiwari A., Majeed, B.,Business Process Analysis and Optimization: Beyond Reengineering, Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on , vol.38, no.1, pp.69-82, 2008.

[5] Blok H.E., Database Optimization Aspects for Information Retrieval. PhD thesis, University of Twente. CTIT Ph.D.-thesis series No. 02-41 ISBN 903651732X, 2002.

[6] Adi-Cristina Mitea, An Optimization Algorithm for Physical Database Design, Proceedings of the 5th WSEAS Int. Conf. on DATA NETWORKS, COMMUNICATIONS & COMPUTERS, pp: 13-18 ,2006.

[7] Rybinkin V., Lukatsky R., An optimization of the structure of databases, 6th WSEAS Int. Conf. on Artificial Intelligence, Knowledge Engineering and Data Bases, pp:147-152, 2007.

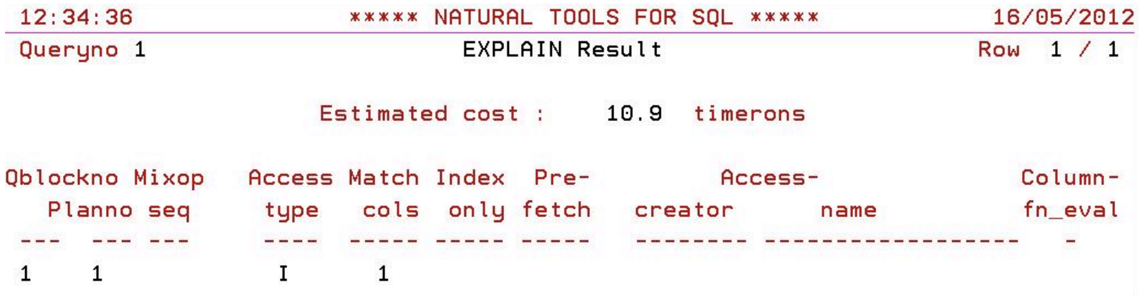


```
12:31:16              ***** NATURAL TOOLS FOR SQL *****          16/05/2012
Queryno 1                          EXPLAIN Result                Row  1 / 2

                     Estimated cost :     50.2  timerons

Qblockno Mixop    Access Match Index  Pre-         Access-             Column-
   Planno seq      type   cols  only fetch   creator      name          fn_eval
 ---   --- ---     ----  ----- ----- -----   -------- -----------------   -
 1     1            I       1
 1     2
```

Figure 3: (Model 1) Cost of accessing entity A.

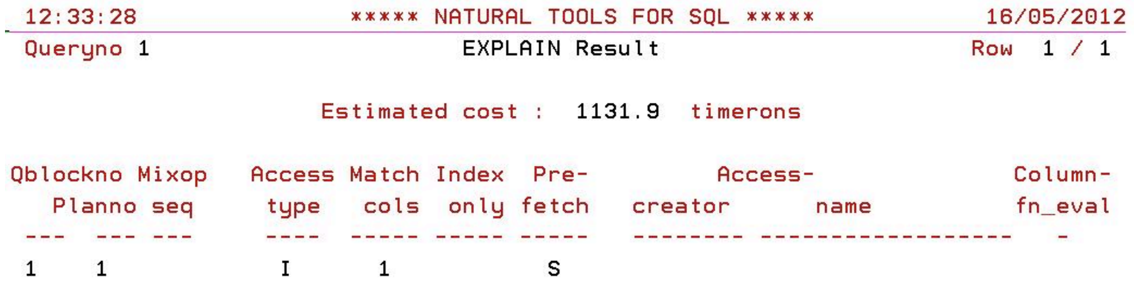


```
12:34:36              ***** NATURAL TOOLS FOR SQL *****          16/05/2012
Queryno 1                          EXPLAIN Result                Row  1 / 1

                     Estimated cost :     10.9  timerons

Qblockno Mixop    Access Match Index  Pre-         Access-             Column-
   Planno seq      type   cols  only fetch   creator      name          fn_eval
 ---   --- ---     ----  ----- ----- -----   -------- -----------------   -
 1     1            I       1
```

Figure 4: (Model 1) Cost of accessing entity B. For each element in entity A, we search all elements that have relation with elements in B.

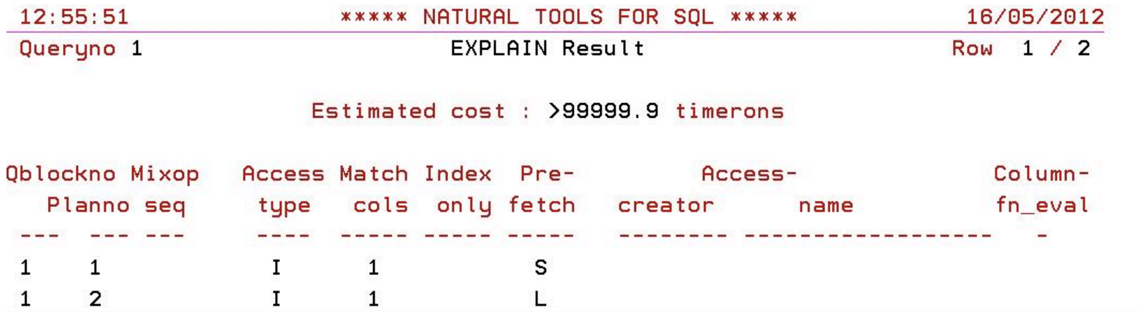


```
12:33:28              ***** NATURAL TOOLS FOR SQL *****          16/05/2012
Queryno 1                          EXPLAIN Result                Row  1 / 1

                     Estimated cost :   1131.9  timerons

Qblockno Mixop    Access Match Index  Pre-         Access-             Column-
   Planno seq      type   cols  only fetch   creator      name          fn_eval
 ---   --- ---     ----  ----- ----- -----   -------- -----------------   -
 1     1            I       1            S
```

Figure 5: (Model 1) Cost of accessing entity C. Starting from B, we search all the elements in entity C that are related to each element in entity B,



```
12:55:51              ***** NATURAL TOOLS FOR SQL *****          16/05/2012
Queryno 1                          EXPLAIN Result                Row  1 / 2

                     Estimated cost : >99999.9 timerons

Qblockno Mixop    Access Match Index  Pre-         Access-             Column-
   Planno seq      type   cols  only fetch   creator      name          fn_eval
 ---   --- ---     ----  ----- ----- -----   -------- -----------------   -
 1     1            I       1            S
 1     2            I       1            L
```

Figure 6: (Model 2) All elements in B that match up with each element in A.

Table 1: Running time of the original process, with the minimum and maximum values

JOBID	STRT DATE/TIME		END DATE/TIME		ELAPSED	CPU	SRB
04227	08/05/12	00:00	08/05/12	02:25	**145.40**	132:14.66	0:00.56
02006	05/05/12	00:00	05/05/12	02:42	162.01	130:37.69	0:00.52
08528	01/05/12	00:00	01/05/12	03:25	205.48	134:48.82	0:00.48
04838	27/04/12	00:00	27/04/12	02:31	150.55	138:20.94	0:00.58
01484	25/04/12	00:43	25/04/12	03:19	155.57	139:07.32	0:00.65
09946	24/04/12	04:01	24/04/12	06:42	161.12	138:18.19	0:00.36
07546	21/04/12	01:44	21/04/12	05:18	**213.18**	140:26.04	0:00.51
05774	20/04/12	00:39	20/04/12	04:00	200.48	139:35.46	0:00.58
04097	19/04/12	02:47	19/04/12	05:30	163.39	137:20.90	0:00.41
02459	18/04/12	03:10	18/04/12	05:50	160.31	139:15.06	0:00.51
00774	17/04/12	01:42	17/04/12	04:46	183.51	154:27.24	0:00.56
08579	14/04/12	01:11	14/04/12	04:34	202.57	139:04.04	0:00.55
06944	13/04/12	01:17	13/04/12	03:52	155.30	137:48.58	0:00.71
05152	12/04/12	00:00	12/04/12	02:33	152.57	137:01.84	0:00.40
03638	11/04/12	00:00	11/04/12	02:31	151.43	137:42.96	0:00.53
02351	10/04/12	03:31	10/04/12	06:11	159.36	138:21.96	0:00.63

Table 2: Unload of the first view, with 4.448.590 items

JOBID	STRT DATE/TIME		END DATE/TIME		ELAPSED	CPU
00923	12/05/12	00:00	12/05/12	00:12	12.31	6:55.65
09083	11/05/12	00:00	11/05/12	00:12	12.31	6:49.61
07396	10/05/12	00:00	10/05/12	00:19	19.54	6:47.77
05858	09/05/12	00:00	09/05/12	00:11	11.39	7:03.88

Table 3: Unload of the second view (historic data), with 6.059.739 items

JOBID	STRT DATE/TIME		END DATE/TIME		ELAPSED	CPU
00924	12/05/12	00:00	12/05/12	00:12	12.36	7:02.20
09091	11/05/12	00:00	11/05/12	00:13	13.01	7:04.05
07397	10/05/12	00:00	10/05/12	00:20	**20.18**	7:08.29
05859	09/05/12	00:00	09/05/12	00:12	12.54	7:13.94

Table 4: Minimum and maximum running time of the implemented process (take into account that it is necessary to add the unload time, refer to table 3)

JOBID	STRT DATE/TIME		END DATE/TIME		ELAPSED	CPU	SRB
00930	12/05/12	00:12	12/05/12	00:18	5.25	4:52.13	0:00.49
09100	11/05/12	00:13	11/05/12	00:18	5.19	4:50.01	0:00.47
07403	10/05/12	00:20	10/05/12	00:25	**5.15**	4:49.64	0:00.50
05997	09/05/12	01:13	09/05/12	01:20	**7.00**	4:59.11	0:00.59

Table 5: Minimum and maximum running time of the original process

JOBID	STRT DATE/TIME		END DATE/TIME		ELAPSED	CPU	SRB
04227	08/05/12	00:00	08/05/12	02:25	**145.40**	132:14.66	0:00.56
07546	21/04/12	01:44	21/04/12	05:18	**213.18**	140:26.04	0:00.51