About the Performance of Fuzzy Querying Systems

Ana Aguilera[2,3], José Tomás Cadenas[1,2], and Leonid Tineo[1,2]

[1] Departamento de Computación, Universidad Simón Bolívar
Caracas, Venezuela
jtcadenas@usb.ve, leonid@usb.ve
[2] Centro de Análisis, Modelado y Tratamiento de Datos, CAMYTD
Facultad de Ciencias y Tecnología, Universidad de Carabobo, Venezuela
[3] Departamento de Computación, Universidad de Carabobo
Valencia, Venezuela
aaguilef@uc.edu.ve

Abstract. Traditional database systems suffer of rigidity. Use of fuzzy sets has been proposed for solving it. Nevertheless, there is certain resistance to adopt this, due the presumption that it adds undesired costs that worsen the performance and scalability of software systems. RDBMS are rather complex by themselves. Extensions for providing higher facilities, with a permissible performance and good scalability would be appreciated. In this paper, we achieve a formal statistics study of fuzzy querying performance. We considered two querying systems: SQLfi (loose coupling) and PostgreSQLf (tight coupling). Observed times for the later are very reasonable. It shows that it is possible to build high performance fuzzy querying systems.

Keywords: Fuzzy Query Processing, PostgreSQL, SQLf, Performance Analysis.

1 Introduction

Emerging applications require DBMS with uncertainty capabilities. SQLf [1] is a fuzzy query language that allows the use of a fuzzy condition anywhere SQL allows a Boolean. SQLf definition was updated till standard SQL:2003 [5]. High performance is a necessary precondition for the acceptance of such systems by end users. However, performance issues have been quite neglected in research on fuzzy database so far. In this paper we deal just with SQLf basic block queries using conjunctive fuzzy conditions, interpreting AND connector by the t-norm minimum. We prove that an implementation with tight-coupling strategy has better performance than one with loose-coupling. Section 2 is devoted to the Performance Analysis and section 3 points out Concluding Remarks and Future Works

2 Performance Analysis

Fuzzy querying systems may be built as extension of existing RDBMS using one of three possible extension methods: loose, middle, and tight-coupling strategies [6].

P. Herrero et al. (Eds.): OTM 2012 Workshops, LNCS 7567, pp. 681–684, 2012.

Loose-coupling consists in implement a logic layer on top of a RDBMS, processing fuzzy queries over the result of regular ones. That's the case of SQLfi [4] where, in order to keep low the extra added cost of external processing, we apply the Derivation Principle [2]. Main advantage of loose-coupling is portability. Open source RDBMS allow tight-coupling. We have made an extension of PostgreSQL for fuzzy query processing, called PostgreSQLf [3]. Fuzzy predicates definitions are stored in a catalogue table. Dynamic structures parse tree, query tree and plan tree were extended for fuzzy queries. The Planner/Optimizer derives Boolean condition from the fuzzy one. The optimization is done with the basis of the derived condition; nevertheless the evaluation is made with the fuzzy one. Evaluator's physical access mechanisms and operators were extended for computing membership degrees.

We have made an experimental performance study using formal model statistic method. The idea is to explain the influence of several factors in the observed values from experiments [7]. The importance of a factor is measured by the proportion of the total variation in the response that is explained by the factor. We designed a multi-factorial experiment of five factors with two levels for each: X1 coupling strategy (loose–tight), X2 volume (low–high), X3 number of tables (single–multiple), X4 number of monotonous fuzzy predicates (single–multiple) and X5 number of unimodal fuzzy predicates (single–multiple). It gives 2^5 combinations of levels, but we replicate, thus we did 64 runs. We took as observed variable of this experiment, the total spent time. This design supposes the value of the answer variable to be a linear combination of the factors and theirs interactions

An experimental database was created with STUDENT - ENROLL – COURSE tables. We populated them with Universidad Simón Bolívar graduate students real data from 1969 to 2007; 20,102 students; 6,640 courses and 1,759,819 enroll rows. We take this population as the low level volume. Remark we use real life data provided by the students control office. We projected a high level data volume with: 100,510 student rows; 6,640 course rows and 5,243,292 enroll rows. To guarantee the same conditions we restart the PostgreSQL server before each experiment. We compute the analysis of variance ANOVA for the model with experiment result. Obtaining that proposed model fits very well. Interaction between volume (X2) and number of unimodal predicates (X5) results been the less significant. On the other hand, experiment shows that factor coupling strategy (X1) an its interactions have very important influence in observed run times. It verifies our working hypothesis. Formal model statistic method experimentally confirms the influence of coupling strategy in run time for fuzzy query processing. In Fig. 1 we may observe the interaction plots between factor coupling strategy and others. Difference between loose-coupling and tight-coupling is evident. Query processing times for loose-coupling strategy is considerably greater than that of tight-coupling. Furthermore, this later seems to rest near to be constant. The run time is inversely proportional to the number of predicates. This is because, we experimented with fuzzy conjunctive queries. Use of more predicates reduces the number of resulting rows, so that also declines the efforts due to computation of membership degrees. Result is as expected.

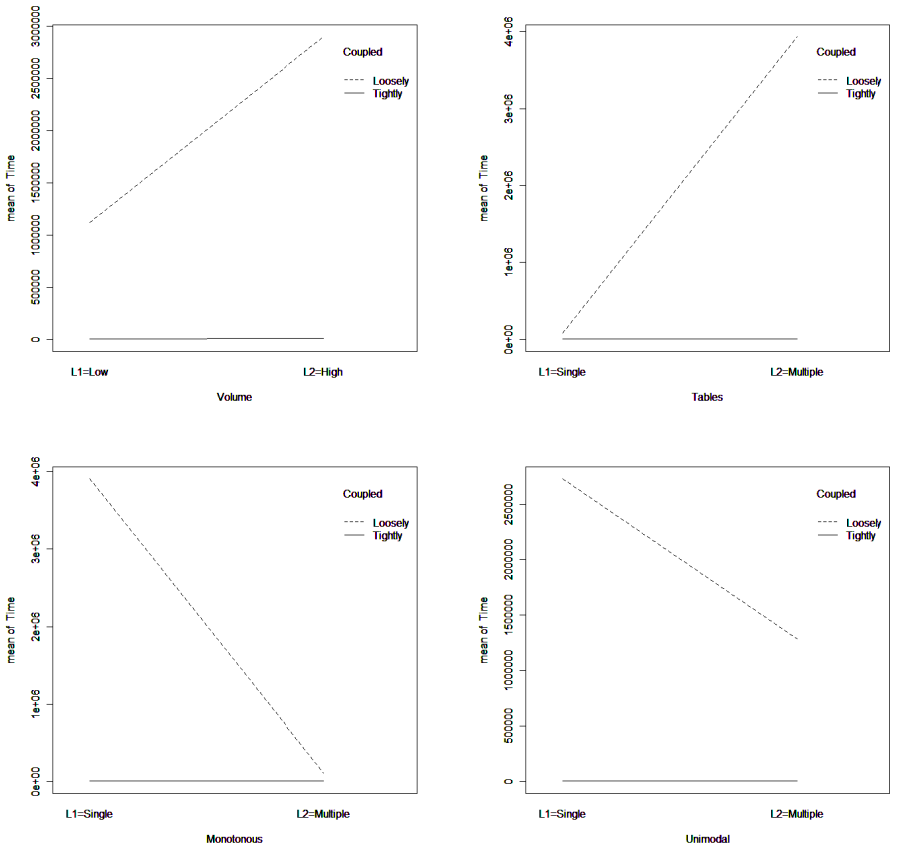

Fig. 1. Interaction Plots for factor Strategy vs: Volume (top-left). Number of Tables (top-right), Monotonous Fuzzy Predicates (bottom-left) and Unimodal Fuzzy Predicates (Bottom- right).

3 Concluding Remarks and Future Works

There is an increasing interest in the problem of dealing with vagueness and uncertainty in data and querying. These systems promises to have much applications. Therefore is very important to provide fuzzy database systems with reasonable performance. In this sense, SQLfi and PostgreSQLf were conceived. The first uses a loose-coupling strategy architecture while the later uses tight-coupling. Experiment shows that factor coupling strategy has very important influence in query processing time. In general, the results indicate that the behavior in run time of tight-coupling is significantly better. We show that between two approaches, the run time difference increases in the extent that database volume does. Moreover, this difference increases in the extent that the volume of rows rises. Observed times for tight-coupling strategy are very reasonable. They are on the order of magnitude between 10^{-1} and 10^{0} in the

scale of seconds. It shows that it is possible to build high performance fuzzy querying systems. In a loose-coupling strategy, the fuzzy query is processed by an external layer. This tier obtains form the RDBMS the result set of a traditional query. This set of rows is rescanned in order to compute satisfaction degree of fuzzy conditions. This increases the query processing cost in terms of total spent time. On the other hand, with tight-coupling strategy, we implement physical access mechanisms that compute membership degrees of fuzzy conditions in the fly. That is while rows are retrieved. Also physical operators are programmed to take into account the fact that row might be provided of membership degree. These operators might combine such degrees according to fuzzy logic conditions involved in the query. It is worth the effort in the implementation of an RDBMS kernel to improve performance and scalability in fuzzy queries. This adds flexibility to SQL improving significantly the run times compared to loose-coupling strategy. In future works, it would be important to analyze how to provide SQLf affects the cost model of the RDBMS optimizer. As current version of PostgreSQLf supports a subset of SQLf, it is recommended to continue the extension in the kernel in order to provide all features form current SQLf definition, which is feasible and promising.

Acknowledgment. We give thanks all people who worked in the development of SQLfi and PortgreSQLf fuzzy querying systems, as well as Venezuela national founds for sciences FONACIT that has supported these developments. This work is done for the glory of the Lord "The horse is made ready for the day of battle, but victory rests with the Lord." Proverbs 21:31 New International Version (NIV).

References

1. Bosc, P., Pivert, O.: SQLf: A Relational Database Language for Fuzzy Quering. IEEE Transactions on Fuzzy Systems 3(1) (February1995)
2. Bosc, P., Pivert, O.: SQLf query functionality on top of a regular relational DBMS. In: Pons, O., Vila, M.A., Kacprzyk, J. (eds.) Knowledge Management in Fuzzy Databases (2000)
3. Aguilera, A., Cadenas, J., Tineo, L.: Fuzzy Querying Capability at Core of a RDBMS. In: Yan, L., Ma, Z. (eds.) Advanced Database Query Systems: Techniques, Applications and Technologies, pp. 160–184. IGI Global, New York (2011)
4. Goncalves, M., Tineo, L.: SQLfi and its Applications. Avances en Sistemas e Informática, vol. 5(2) (2008); Medellin, ISSN 1657-7663
5. Goncalves, M., González, C., Tineo, L.: A New Upgrade to SQLf: Towards a Standard in Fuzzy Databases. In: Proc of DEXA 2009 Workshops (2009)
6. Timarán, R.: Arquitecturas de Integración del Proceso de Descubrimiento de Conocimiento con Sistemas de Gestión de Bases de Datos: un Estado del Arte. Ingeniería y Competitividad 3(2) (2001)
7. Raj, J.: The Art of Computer Systems Performance. John Wiley/Sons, Inc. (1991)