Preferred Skyline: A Hybrid Approach Between SQLf and Skyline

Marlene Goncalves and María-Esther Vidal

Universidad Simón Bolívar, Departamento de Computación, Apartado 89000,
Caracas 1080-A, Venezuela
{mgoncalves, mvidal}@usb.ve

Abstract. The World Wide Web (WWW) is a great repository of data and it may reference thousands of data sources for almost any knowledge domain. Users frequently access sources to query information and may be interested only in the top k answers that meet their preferences. Although, many declarative languages have been defined to express WWW queries, the problem of specifying user preferences and considering this information during query optimization and evaluation remains open. Most used query languages, such as SQL and XQUERY, do not allow specifying general conditions on user preferences. For example, using the SQL ORDER BY clause one can express the order in which the answer will appear, but the user must be aware of filtering the tuples that satisfy their preferences. Skyline and SQLf are two extensions of SQL defined to express general user preferences. Skyline offers physical operators to construct a Pareto Curve of the non-dominated answers. Skyline may return answers with bad values for some criteria and does not discriminate between the non-dominated answers. On the other hand, SQLf gives the best answers in terms of user preferences, but it may return dominated answers. Finally, the skyline operator evaluation time is higher than SQLf. We proposed a hybrid approach, Preferred Skyline, integrating skyline and SQLf to produce only answers in the Pareto Curve with best satisfaction degrees. We report initial experimental results on execution time and answer precision. They show that Preferred Skyline consumes less time than Skyline and its precision is higher than SQLf.

1 Introduction

The World Wide Web (WWW) is a great repository of data and it may reference thousands of data sources for almost any knowledge domain. Constantly, new data sources are published, increasing dramatically the size of the WWW.

Users frequently access sources in order to query information and may be interested only in answers that satisfy their preferences. Electronic commerce is an example of this type of searching, where users express their preference criteria in terms of the quality of services/products.

Although, many declarative languages have been defined to express WWW queries, the problem of specifying user preferences and considering this information during query optimization and evaluation remains open. On one hand, most used query languages, such as SQL (Structured Query Language) [1] and XQUERY [17], do not allow to specify general conditions on user preferences. For example, one can express the order in which the answer of a SQL query will be produced by using the ORDER BY clause. However, to choose the ones that satisfy the user preferences, a filter process must be run on the top of the answer. This process becomes more complex when user criteria involve three or more conditions.

Skyline [9] and SQLf [5] are two SQL extensions to express user-preferences. Skyline orders the answers of a query in terms of several criteria and outputs the set of points that are not dominated by any other point in the dataset. This set is called skyline or first stratum in the Pareto curve. SQLf is an approach defined to express fuzzy queries; user preferences are defined as utility functions. These functions are used to compute the membership degrees of the tuples in the answer of a fuzzy query.

The SQLf query algorithm is based on the Derivation Principle. Following the Derivation Principle a fuzzy query is rewritten into a regular query called derived query. The derived query evaluation time is lower than fuzzy query processing, which is the main problem for fuzzy databases [4]. For the best of our knowledge, there have not defined fuzzy query optimization and evaluation techniques. On the other hand, a naive skyline evaluation is excessively expensive and efficient physical operators have been defined. SFS (Sort Filter Skyline) [9] is an efficient algorithm that allocates skyline answers in a *window*; tuples in the answer are chosen from the input table, which is sorted in terms of the user-preference attributes.

The main objective of skyline algorithms is to efficiently identify optimal results in the space of solutions that meet the multiple user criteria. The solution of a multi-objective optimization problem is referred to as the Pareto curve [14]. A Pareto curve has an exponential number of points and it may be impossible to build this curve in reasonable time. For this reason, algorithms find approximations to Pareto optimal. Skyline identifies the first point in an approximation of the Pareto curve. Elements in the first point, are non-dominated by any other element. An element dominates another element if it is no worse in all criteria, and better in at least one criterion. SQLf may return dominated elements and may not identify all the non-dominated. Filtering dominated answers in SQLf will allow one to find better answers in terms of user preferences. On the other, Skyline can return answers with bad values for some criterion, i.e., it identifies the answers with better values in at least one criterion and worse values in the rest. Finally, Skyline does not discriminate between the non-dominated answers and SQLf does discriminate them in terms of a user-defined predicate.

We propose a hybrid approach, Preferred Skyline, integrating Skyline and SQLf in order to produce only answers in the Pareto Curve that satisfy user criteria. We report initial experimental results on execution time and answer precision. Our initial results show that Preferred Skyline consumes less time than Skyline and its precision is higher than SQLf.

2 Background and Related Work

Kung et al. proposed the first Skyline algorithm in [13], referred to as the maximum vector problem and it is based on the divide & conquer principle. Progress has been made as of recent on how to compute efficiently such queries in a relational system and over large datasets [2][9][16]. In [2], the Skyline operator is introduced. They proposed two algorithms: a block-nested loops style algorithm (and variations) and a divide-and-conquer approach derived from work in [13][16]. In [16], an algorithm that uses specialized indexing for Skyline evaluation is introduced. In [9], a general Skyline algorithm was developed and it is based on the "block-nested loop" algorithm of [2] that is faster, pipelinable and more amenable.

On the other hand, several efforts have been made by different authors in order to provide flexible querying database systems. Bosc and Pivert [5] have proposed SQLf, a fuzzy-set-based SQL extension. In [11][12] SQLf2 and SQLf3 are defined as an SQLf extension with the features of SQL2 and SQL3. There are some query processing mechanism [3][4][6][7] and the most relevant is derivation principle [6][7] because it has a less cost.

2.1 Skyline Query Language

An operator, named Skyline, is defined in [2]. In Fig. 1, SQL is extended with a SKYLINE OF clause.

```
SELECT <attributes> FROM <relations> WHERE <conditions>
      GROUP BY <attributes> HAVING <conditions>
   SKYLINE OF a1 [min|max|diff],…,an [min|max|diff]
```

Fig. 1. A proposed skyline operator for SQL

A SKYLINE OF clause body represents a list of attributes or Skyline dimensions used to rank the dataset. Each dimension can be an integer, float, or a date and may be annotated with the directives: min, max and diff. Mix and max indicate minimum or maximum values and the diff directive defines the interest in retaining the best choices with respect to every distinct value of the attribute. Finally, if none directive is stated, max is the default.

SFS (Sort-Filter-Skyline) [9] is an efficient algorithm that improves the performance of a skyline time computation. First, this algorithm sorts the input table and then, it passes over the sorted tuples to identify the ones that are non-dominated. The main disadvantage of this approach is that it first needs to order the input table and then, scan it to check the dominance relationship.

2.2 SQLf Query Language

One remarkable effort in the creation of flexible querying system for relational databases is SQLf [5]. Fuzzy queries involve fuzzy terms (atomic predicates, modifiers, connectors, comparators and quantifiers) whose semantic is user and context depended. Fig. 2 shows an SQLf basic query block.

```
          SELECT <attributes> FROM <relations>
WHERE <fuzzy conditions> WITH CALIBRATION [n|α|n,α]
```

Fig. 2. SQLf basic block

An SQLf query answer corresponds to the tuples in the Cartesian product of the relations in the FROM clause that satisfy the fuzzy condition. These tuples consist of the attributes in the SELECT clause. Some fuzzy logical expressions can be used with user-defined terms and predefined operators in the WHERE clause. A tolerance is specified in the CALIBRATION clause and indicates the condition to be satisfied by the best tuples. The process to identify the best tuples is called calibration. Two calibration types have been proposed: Quantitative and Qualitative. In case of Quantitative calibration, a maximum number "n" of answers is obtained. On the other hand, tuples whose membership value is greater or equal to "α" are produced if the calibration is Qualitative.

An efficient SQLf evaluation mechanism is based on the Derivation Principle [6][15]. The Derivation Principle Strategy attempts to keep low the number of row access in fuzzy query processing by means of the distribution of the α-cut over conditions involved in the fuzzy query. It derives a regular SQL query whose result contains the rows satisfying the fuzzy query with a degree value of at least α. This principle may be used for the evaluation of fuzzy queries avoiding the scanning of the whole database. A derived necessary condition is a basic concept in the Derivation Principle application. It is a regular condition that expresses the α-cut of a fuzzy condition. It is denoted by: *DNC(<fuzzy condition>, ≥, α).*

Consider the following example to illustrate the Derivation Principle. Suppose *location* is a relation with an attribute *priority* and, *BestZone* is a fuzzy predicate on *priority* domain that measures the *goodness* of a zone. Fig 3 presents the definition of the *BestZone* function. The query: *Find those locations in best areas with a tolerance level of 0.5* is expressed using SQLf as follows:

```
SELECT * FROM location WHERE priority = BestZone
                WITH CALIBRATION 0.5
```

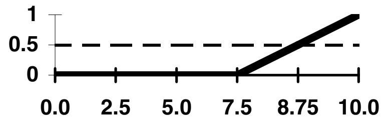

Fig. 3. *BestZone* predicate definition by means of its membership function. We remark the satisfaction 0.5 that is used as a threshold in the example.

First, applying the Derivation Principle, the derived necessary condition: *DNC(priority = BestZone, ≥, 0.5) = Priority ≥ 8.75)* is obtained. Since, the *BestZone* fuzzy function indicates that all the locations whose priority values is greater or equal to 8.75 will have a membership value equal to 0.5, then, the initial query is translated into the following:

```
SELECT * FROM location WHERE priority ≥ 8.75.
```

3 Motivating Example

Consider the following relational table that represents a restaurant guide: Guide(idRest,Reviewer,AvegPrice,QService,QFood,QPlace).

A restaurant is described by an identifier (idRest), the reviewer name (Reviewer), the average of the dish price (AvegPrice), and three values between 1 to 30 that measure the quality of the service (QService), the quality of food (QFood) and the quality of the place (QPlace), respectively.

Suppose the top K restaurants will receive an award, so tuples in table Guide must be ranked in terms of the values of: QService, QFood, QPlace and AvegPrice. A restaurant can be postulated for the award if and only if there is no other restaurant with lower average price and better quality of service, food and place. To nominate a restaurant, one must identify the set of all the restaurants that are not dominated by any other restaurant in terms of these criteria. Winners will be selected among nominates in terms of the top K values of an overall preference that combines values of the previous attributes weighted by a confidence degree in the reviewer decision.

Skyline and SQLf are two extensions of SQL language that could be used to identify the winners from table Guide. However, none of them will provide the set of winners and post-processing will be needed to filter the winners.

Skyline offers a set of operators to build an approximation of a Pareto curve (strata) or set of points that are not dominated by any other point in the dataset (skyline or first stratum). Thus, by using Skyline, ones could just obtain the nominated restaurants.

On the hand, SQLf will allow referees to implement a score function and filter some of the winners in terms of this function. In order to choose winner restaurants, SQLf computes the score to each tuple without checking dominance relationship between tuples in the dataset. Finally, also top K query approaches rank a set of tuples according to some provided functions and do not check dominance relationship.

Although solutions produces by either SQLf or top K queries, will correspond to winners, some nominates may not be considered and in consequence some winner may not be identified [8].

To identify the set of all the winners, a hybrid approach that combines the goodness of Skyline and SQLf or top K will be needed.

In this paper, we propose a Preferred Skyline approach that filters among an approximation of the Pareto curve, the points that meet user preference criteria the most.

4 Preferred Skyline

Preferred Skyline is a hybrid approach between Skyline and SQLf, that offers the best points of the first stratum or skyline of Pareto Curve based on a quantitative or qualitative calibration.

We define its basic block as in Fig. 4. A Preferred Skyline query answer corresponds to the tuples on the Cartesian product of the relations in the FROM clause that are non dominated by any other tuple according to fuzzy skyline criteriaand whose score on a fuzzy condition function, is greater or equal than α.

A tuple t is dominated by a tuple t', if and only if, t' is better than t as in the score function as in all fuzzy skyline criteria. If “n” is specified in the WITH CALIBRATION clause, the n best skyline tuples with the highest score are returned.

```
SELECT <attributes> FROM <relations>
WHERE <fuzzy_conditions>
SKYLINE OF <fuzzy_skyline_criteria> WITH CALIBRATION [n | αln,α]
```

Fig. 4. SQLf basic block

We have implemented Preferred Skyline as a two steps process that combines SQLf and Skyline. In the first step, tuples in the input tables are filtered using the SQLf Derivation Principle-based algorithm. Then, in the second step, the SFS algorithm identifies non-dominated tuples. Dominated tuples are computed in terms of the fuzzy criteria and the fuzzy predicates calculated in the previous step. Note that this implementation is sound but it may not complete, i.e., there may be some non-identified nominates.

Preferred Skyline algorithm has a (worst-case) complexity of $O(m(\log m)^{d-2})$, where “m” the number of points identified in the first step, and “d” the number of skyline dimensions [13]. The Preferred Skyline time complexity bound is based on the time complexity bounds of SQLf algorithms and Skyline. First, the SQLf Derivation Principle algorithm requires to scan a subset of the input that meets the fuzzy predicates. Then, running time reflects the database access time plus tuple processing time. The database access time is “c_1” times “m” with “c_1” a constant, “m” the number of returned points, “n” the number of points in the data set, $m \leq n$. The tuple processing time is “c_2” times “m” with “c_2” a constant. A time complexity bound to compute the fuzzy set is $O(m)$ [15]. This previous filtering reduces Skyline algorithm time. The Skyline algorithm running discards solutions that are not better across all the criteria and are produced by SQLf. The running time of Skyline algorithms can be very high and it can be shown that the best Skyline algorithm to compute an approximation of a Pareto curve has a (worst-case) complexity of $O(n(\log n)^{d-2})$, with “n” the number of points in the data set and d the number of dimensions [13].

5 Initial Experimental Study

We compare the quality of the answers produced by Preferred Skyline, Skyline and SQLf, and their performances. We implemented following three algorithms: the basic SFS algorithm without optimizations [9], the Principle-Derivation-based algorithm [4] and Preferred Skyline as the combination of the previous ones. All the algorithms were implemented in PL/SQL and Swi Prolog and executed on a single processor machine: Intel 866-MHz PC with 512-MB main memory and an 18-GB disk running Red Hat Linux 8.0. In this initial study, we run experiments over synthetic databases.

5.1 Experiment 1: Quality of Preferred Skyline wrt SQLf and Skyline

We study the quality of the answers identified by Preferred Skyline, Skyline, SQLf with respect to the top K answers (TopK). We run nine medium and large queries. Queries represented in SQLf, Skyline and Preferred Skyline languages were defined over the same table and attributes. SQLf queries were randomly generated and characterized by the following properties: (a) there was only one table in the FROM clause; (b) attributes in the fuzzy predicates were randomly chosen among the table attributes using a uniform distribution; (c) base fuzzy predicates were defined on the basis of the fuzzy predicate minimum which is defined in Fig. 5; (d) a fuzzy condition in the WHERE clause was a conjunction of five or nine base fuzzy conditions and (e) the quality calibration was 0.75. Skyline queries had the following properties: (a) there was only one table in the FROM clause; (b) the Skyline directive was min. Preferred Skyline queries had the SQLf query properties plus the Skyline directive min. The size of the table varied between 100,000 and 1,000,000 tuples. Each tuple had ten integer values in the range from 1 to 30 and one string value. Each tuple was randomly generated following a uniform distribution.

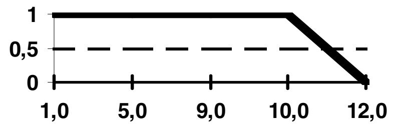

Fig. 5. *Minimum* predicate definition by means of its membership function

In Table 1, we report the following results for the nine randomly generated queries.

— The average number of tuples in the answers produced by Preferred Skyline, Skyline and SQLf.

— The average number of additional tuples identified by SQLf and Skyline with respect to Preferred Skyline.

In Table 1 we can observe that for queries with five fuzzy conditions SQLf returns many additional tuples either for the small or large database. Almost all of these tuples are dominated. This may be because first the SQLf preference criteria are based on Fuzzy Logic and they are more relaxed than Boolean criteria and second, SQLf does not able to check the dominance relationship directly. On the hand, Skyline only returns the tuples in the skyline, i.e., non- dominated tuples, and in consequence, it is more precise.

Finally, for queries with nine fuzzy conditions, SQLf returns few or none additional tuples in all cases. SQLf is more precise maybe because the fuzzy condition criteria are more selective. However, Skyline returns more tuples, many of them have bad values in some criteria. In consequence, Skyline answers are less precise.

Table 1. Small database and five dimensions

Database	Dimensions	Preferred Skyline	SQLf	Skyline	SQLf Additional Tuples	Skyline Additional Tuples
Small	5	40	505	107	465	67
Large	5	32	7850	39	7818	7
Small	9	7	7	8350	0	8344
Large	9	126	166	18311	40	18185

5.2 Experiment 2: Performance of Preferred Skyline wrt SQLf and Skyline

We report time evaluation for nine randomly generated queries expressed by using Preferred Skyline, Skyline, SQLf languages. A fuzzy condition in the WHERE clause is either a conjunction of one, five or nine base fuzzy conditions. The rest of the parameters are set as in experiment 1.

First, we suppose that SQLf queries's execution time is lower than Skyline's because Skyline requires to find all tuples while, SQLf just needs to identify a subset of them. This is because a query that filters the tuples that do not satisfy the tolerance is obtained as result of the application of the Principle of Derivation.

In Fig. 6 we report the effect of the number of criteria in the performance of Skyline, SQLf and Preferred Skyline. Criteria are referred as Skyline dimensions or fuzzy conditions. Time units are seconds and the results are normalized by means of log function. We can observe the following:

a) The processing time for SQLf decreases as the number of fuzzy conditions increases. This may be because the conditions became more selective and the number of filtered tuples decreases.

b) There is not correlation between Skyline evaluation time and the number of dimensions. However, we can observe that the Skyline time increases as the number of tuples in the skyline increases. For all dimensions, Skyline time is higher than the SQLf time. It may be because Skyline scans all the tuples and checks the dominance relationship. For nine dimensions, the skyline is the largest set and the Skyline time is the highest.

c) Similar to the SQLf behavior, the Preferred Skyline evaluation time decreases as the number of dimensions increases. Moreover, the Preferred Skyline evaluation time is lower than Skyline's and higher than the SQLf's. This may because the Preferred Skyline constructs the Pareto curve on top of the set of tuples produced by the SQLf, instead of considering the whole initial set of tuples.

In Fig. 7 we report the effect of the database size on the Skyline, SQLf and Preferred Skyline evaluation time. We can observe that the processing time for Skyline, SQLf and Preferred Skyline decreases as the size of the database increases. Moreover, the Preferred Skyline evaluation time is lower than Skyline's and higher than the SQLf's.

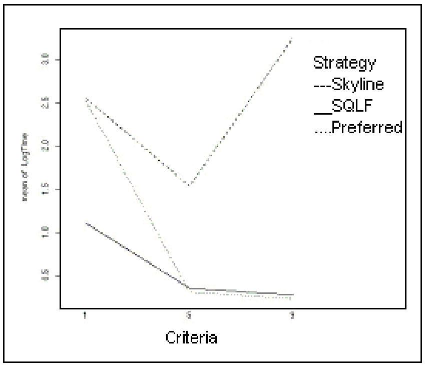

Fig. 6. Dimension-Strategy Interaction Influence in Time for Basic Block Queries

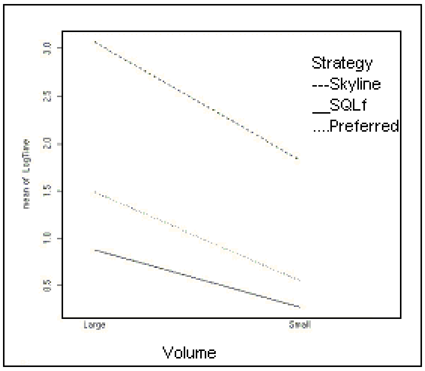

Fig. 7. Volume-Strategy Interaction Influence in Time for Basic Block Queries

6 Conclusions

We have defined a new approach called Preferred Skyline that integrates Skyline and SQLf approaches and returns the best answers of the skyline (first stratum).

We study the performance and quality of Preferred Skyline wrt. Skyline and SQLf. The initial experiments show that Preferred Skyline is more precise than either Skyline and SQLf and its evaluation time is lower than Skyline's and higher than SQLf's. On the other hand, Preferred Skyline may not identify some relevant answers. Additionally, Preferred Skyline is not able to produce just the top k answers when the skyline cardinality is greater than k.

In order to improve the Preferred Skyline and reduce its limitations, it should be extended with the capabilities of finding possible answers in the following strata or next points in the Pareto curve. Necessary probe criteria are needed to ensure that only the required query condition evaluations are performed. In addition, to efficiently evaluate Preferred Skyline queries, a physical operator and cost model must be defined and incorporated into an existing database management system. In the future, we plan to define physical Preferred Skyline operators, their cost models and integrate them into PostgreSQL database management system.

References

1. ANSI X3. "American National Standard for Information Systems: Database Language SQL". American National Standards Institute, NY, (1986), 135.
2. Börzsönyi, S., Kossmann, D. and Stocker, K. "The skyline operator". Proceedings of International Conference on Data Engineering, (2001), 421-430.
3. Bosc, P. and Brisson, A. "On the evaluation of some SQLf nested queries". Proceeding International Workshop on Fuzzy Databases and Information Retrieval, (1995).

4. Bosc, P. and Pivert, O. “On the efficiency of the alpha-cut distribution method to evaluate simple fuzzy relational queries”. Advances in Fuzzy Systems-Applications and Theory, (1995), 251-260.
5. Bosc, P. and Pivert, O. “SQLf: A Relational Database Language for Fuzzy Querying”. IEEE Transactions on Fuzzy Systems 3, 1 (Feb 1995).
6. Bosc, P. and Pivert, O. “SQLf Query Functionality on Top of a Regular Relational Database Management System”. Knowledge Management in Fuzzy Databases (2000), 171-190.
7. Bosc, P., Pivert, O. and Farquhar, K. “Integrating Fuzzy Queries into an Existing Database Management System: An Example”. International Journal of Intelligent Systems 9, (1994), 475-492.
8. Chang, K. and Hwang, S-W. “Minimal Probing: Supporting Expensive Predicates for Top-k Queries”. Proceedings of the ACM SIGMOD Conference, (Jun 2002).
9. Chomicky, J., Godfrey , P. Gryz, J. and Liang, D. “Skyline with Presorting”. Proceedings of 19th International Conference on Data Engineering**,** (Mar. 2003).
10. Florescu, D., Levy, A., Manolescu, I., Suciu, D. “Query Optimization in the Presence of Limited Access Patterns”. Proceedings of the ACM SIGMOD Conference, 1999.
11. Goncalves, M. and Tineo, L. “SQLf: Flexible Querying Language Extension by means of the norm SQL2”. The 10th IEEE International Conference on Fuzzy Systems 1, (Dec. 2001)
12. Goncalves, M. and Tineo, L. “SQLf3: An extension of SQLf with SQL3 features”. The 10th IEEE International Conference on Fuzzy Systems 1, (Dec. 2001)
13. Kung, H. T., Luccio, F. and Preparata, F. P. “On finding the maxima of a set of vectors”. Journal of the ACM, 22 (4), 1975.
14. Papadimitriou, C. H. and Yannakakis, M. “Multiobjective Query Optimization”. Proc. ACM SIGMOD/SIGACT Conf. Princ. Of Database Syst. (PODS), Santa Barbara, CA, USA, May 2001.
15. Pivert, O. “Contribution à l’Interrogation Flexible de Bases de Données: Expression et Évaluation de Requêtes Floues”. Thèse de Doctoract, Université de Rennes I, France, 1991.
16. Preparata, F. P. and Shamos, M. I. “Computational Geometry: An Introduction”. Springer-Verlag, 1985.
17. XQUERY 1.0: An XML Query Language. Available at http://www.w3.org/TR/xquery