Research Journal of Applied Sciences, Engineering and Technology 11(3): 267-273, 2015
DOI: 10.19026/rjaset.11.1716
ISSN: 2040-7459; e-ISSN: 2040-7467

Submitted: December 10, 2014 Accepted: February 5, 2015 Published: September 25, 2015

Research Article

Analysis of Data Mining Dataset Using Fuzzy Based Unnested Select SQL Queries

[1]D. Veni and [2]K.R. Chandran
[1]Anna University,
[2]Information Technology Department, PSG College of Technology, Coimbatore, Tamilnadu, India

Abstract: The aim of this study is to improve the existing traditional databases. Some new techniques have been involved to handle the imprecise or uncertain information from the dataset. Dataset is prepared and used to analyze the data mining project which consume more time and need many complex queries. Nested query is predominant method to handle complex queries. Execution of a nested query may cause the heavy performance penalty. The main objective of this study is to reduce the heavy performance penalty of nested queries by using the unnested queries. The unnested queries produce the equivalent output as nested queries with minimum penalty and execution time. Success of unnested queries are examined using join algorithms. It is more efficient than the nested-loop algorithms which are used to evaluate the nested queries. In this study, unnested queries are used to analysis the data-mining project in dataset, we get the result from combining fuzzy set theory. In experimental results, we have shown that the performance of evaluating the unnesting techniques with extended merge-join and horizontal aggregations techniques CASE, SPJ and PIVOT in dataset. Thus, unnested queries improve the performance of execution and linear scalability.

Keywords: Fuzzy queries, nested queries, PIVOT, set exclusive operator

INTRODUCTION

In data mining, especially in a relational database there is a need of significant effort to prepare a summary data set. Many techniques take data set as input in a horizontal layout, with several records and one variable or dimension per column. That is the case with models like clustering, classification, regression and PCA; consult (Ordonez, 2010). These techniques are usually used to describe the data set.

Data aggregation is a process in which information is collected and expressed in a summary form and which is used for purposes such as statistical analysis. The Structured Query Language (SQL) queries are more flexible and easier to process relational database for data aggregation. There are numerous functions and operators for aggregation using SQL. Generally, aggregation is the summation of columns and return the average, maximum, minimum or row count over groups of rows. This process limits to build large data sets for data mining purposes.

The concept of fuzzy set theory with database technology is an effective technique (Baldwin, 1983) where the database system uses fuzzy based SQL query language that is used here in SQL language. In this, the queries contain fuzzy relation where each tuple satisfies the query condition to extend its membership degree. The query optimization is done by decomposing a complex query into nested query. In standard SQL, the nested queries suffer from heavy performance penalty when they contain complex queries. A common technique to evaluate a nested standard SQL query is to transform the query into an equivalent flat query and then to evaluate the flat query (Qi *et al*., 2001). This type of query is called unnesting technique, which is used by many researchers in relational database systems (Kim, 1982; Lohman *et al*., 1984). This unnested query is processed by different join algorithm and more successful than nested-loop algorithms. Nested query is predominant method to handle complex queries. Execution of a nested query may cause the heavy performance penalty.

The main objective of this study is to reduce the heavy performance penalty of nested queries by using the unnested queries. The unnested queries produce the equivalent output as nested queries with minimum penalty and execution time. This study is focused to examine unnested queries using join algorithms for dataset which is more efficient than the nested-loop algorithms to evaluate the nested queries. These unnested queries are used with fuzzy to analyze the data mining project in dataset and the results are obtained by combining fuzzy set theory. The nested and equivalent unnested queries are explained in detail.

LITERATURE REVIEW

Gray *et al*. (1996) proposed a relational aggregation based operation that generalizing Group-

Corresponding Author: D. Veni, Anna University, Coimbatore, Tamilnadu, India

By, Cross-Tab and Sub-Totals. The cube operator generalizes the cross tab, drill-down, histogram, roll-up and sub-total constructs. These operators are used in more complex non-procedural data analysis programs and data mining. It treats each of the N aggregation attributes of N-space dimension. The aggregate of a particular set of attribute value is a point in this space and the set of points form an N- dimensional cube. By aggregating the N-cube to lower dimensional spaces super-aggregates are computed. Creating a data cube requires generating the power set of the aggregation columns.

Ordonez (2004) proposed two SQL aggregate functions to compute percentages that addressing many limitations. The first function returns one row for each percentage in vertical form and the second function returns each set of percentages adding 100% on the same row in horizontal form. These aggregate functions are used to introduce the concept of percentage queries and to generate efficient SQL code in data mining. Queries using percentage aggregations are called percentage queries. When computing vertical percentage queries two practical issues were identified. First is missing rows and second issue is division by zero.

Wang *et al*. (2003) proposed a Complete SQL Extension for data Mining and Data Streams. This technique is a powerful database language and system that enables users to develop complete data-intensive applications in SQL, by writing new aggregates and table functions in SQL, rather than in procedural languages as in current Object-Relational systems. The "ATLaS" system consists of applications which consist of various data mining. A function is coded in "ATLaS" SQL and execute with a modest performance overhead with respect to the same applications written in C/C++. Using the schema and queries in Query Repository this system handles continuous queries.

Ordonez and Chen (2012) introduced a new class of aggregate functions for a horizontal layout that build data sets using automating SQL query writing and extending SQL capabilities. Horizontal aggregations are evaluated by three fundamental methods namely CASE; SPJ and PIVOT. These three methods produce the same result and performance is analyze by time complexity and I/O cost.

PROPOSED METHODOLOGY

Fuzzy set: An element a Є X is defined in a fuzzy set F if and only if $\mu_F(a)>0$ and to be a full member if and only if $\mu_F(a) = 1$ where A is defined by a membership function $\mu_F()$ which is given to every element a Є X, a membership degree (a) Є [0, 1].

Relational database: In this study, we used following notations in Table 1.

Each attribute has set of data values as its domain where each data values u is associated with a possibility distribution. It has a membership function denoted by μ_u and over domain of attribute. Its possibility distribution of crisp data values is defined by:

Table 1: Notation

Notation	Description
W, X, Y and Z	Relations with upper case letters
w, x, y and z	Tuples with lowercase letters
X.A	The attribute A of the relation X

$$\mu_u(a) = \begin{cases} 1 \; if \; a = u \\ 0 \; otherwise \end{cases} \quad (1)$$

Let P (A) is set of all possibility distributions that may be defined over the domain of an attribute A of relation R with a schema A1,…., An, where Ai is an attribute, is defined as:

X = P (A1) x P (A2) x ……..x P(An) x D

where, D is a system-supplied attribute for membership degree with a domain [0, 1] and x denotes the cross product:

Fuzzy SQL of the statement is specified in
SELECT Attributes
FROM Relations
WHERE condition

The following query is about pairs of male X and female Y persons who have the same age and M has a more than "high" income:

Query 1
SELECT X.No, Y.No
FROM X, Y
WHERE Y.AGE = M.AGE AND
X.INCOME> "high"

Hence, AGE and INCOME may have fuzzy values, comparisons made are fuzzy.

Nested fuzzy queries: The nested Fuzzy SQL query is used to express a complex query. The below example is about the nested Fuzzy SQL query find the number of old female persons who have middle age male person's income:

Query 2
SELECT Y.No
FROM Y
WHERE Y.AGE = ª old° AND
Y.INCOME IN
(SELECT X.INCOME
FROM X
WHERE X.AGE = ªmiddle age°)

This nested query uses nested loop where the outer block contains relation Y and the inner block contains relation X. Execution of this nested query of the inner

relation X is scanned once for every tuple in outer relation Y. Hence it takes more processing cost, I/O cost when the number of tuples in X is very large. In order to avoid the scanning process of every tuple in intermediate relation and speed up the process, use unnesting Query 2 in the conventional databases by obtain the following Query 2':

Query 2'
SELECT Y.NO
FROM Y, X
WHERE Y.AGE = "old" AND
X.AGE = "middle age" AND
Y.INCOME = X.INCOME

Basic nested queries: Two basic nested queries are used, type A and type B. The distinguish between two types is type A query, inner block contains join predicate referencing the outer relation and type B query does not. k1 indicates the conjunction of predicates involving the outer relation where k2 contains only the inner relation.
The following type B of Query B is below:

Query B
SELECT X.A
FROM X
WHERE k1 AND X.B is in
(SELECT Y.C
FROM Y
WHERE k2)

The following unnested Query B' is identical for Query B:

Query B'
SELECT X.A
FROM X, Y
WHERE k1 AND X.B = Y.CAND k2
Query A
SELECT X.A
FROM X
WHERE k1 AND X.B is in
(SELECTY.C
FROM Y
WHERE k2AND Y.V = X.U)

The following unnested Query A' is identical for Query A:

Query A'
SELECT X.A
FROM X,Y
WHERE k1 AND k2AND X.B = Y.C
AND Y.V = X.U

Set exclusion operator: The set inclusion operator "is in" is replaced by the "is not in" in Kim (1982). This Antijoin predicate is used to unnesta type B or type A query and left out predicates p1 and p2 in type B or type A query. Query J is of type A with the set exclusion operator.

Query J
SELECT X.A
FROM X
WHERE X.B is not in
(SELECT Y.C
FROM Y
WHERE Y.V = X.U)

The following query 3 is example of type J and finds the number of employees in Sales department who don't have an income of any employee in Research department with his/her age X and Y.

Query 3
SELECT X.NO
FROM ESALES X
WHERE X.INCOME is not in
(SELECT Y.INCOME
FROM ERESEARCH Y
WHERE X.AGE = Y.AGE)

To unnest Query J, temporary relation uses both the WITH and the GROUPBY clauses and explicitly refers to the membership degrees of X, Y and the answer relations. X. K be a key of X.
Query J can be unnested to the following Query JX':

JT(K, M) = (SELECT X.K, R.E, MIN(D)
FROM X, Y
WHERE X.D AND ¬(Y.D AND
X.B =Y.C AND X.U = Y.V)
WITH D ≥0
GROUPBY X.K)
Query JX'
SELECT M
FROM JT

The query JT is identical and Query J' dispose of attribute K. The impossible directly use of membership degree attributes X.D and Y.D because each predicate is evaluated to a satisfaction degree and the membership degree can also be a satisfaction degree where a membership degree of attribute can used itself as a predicate.

Nested queries with aggregate: The unnesting of a type AA query (Kim, 1982) in which the inner block of query has a join predicate which refer the outer relation and SELECT clause has aggregate function which generate a non-empty values from a nonempty set of values.SQL has aggregate functions such as COUNT, AVG, SUM, MIN and MAX. COUNT returns the number of values in set.

This function can be applied in nested and the unnested queries to the same set of values. The following Query AA is a type AA nested query:

```
Query AA
SELECT X.A
FROM X
WHERE k1 AND X.Boperator1
(SELECTAGG (Y.C)
FROM Y
WHERE k2 AND
Y.V operator2 X.U)
```

Operator1 and operator2 in predicate are comparison operator among {=, ≤, ≥, <,>} and AGG is one of the aggregate functions {MAX, MIN, AVG, SUM and COUNT}. If there is no joinpredicate in the inner block, then the inner block gives the same single value for every tuple in X. Hence, no unnesting is needed.

The following example for type AA query finds the names of cities in region A, each of cities which has an average home-income higher than the maximum average home-income of cities in region B with same population:

```
Query 4
SELECT X.NAME
FROM REGIONA X
WHERE X. A_H_INCOME>
(SELECT MAX (Y.A_H_INCOME)
FROM REGIONB Y
WHERE Y.POPULATION
= X.POPULATION)
```

To unnest Query AA is defined as follows:

```
R1(U) = (SELECT X.U
FROM X
WHERE k1)
R2(U,A). = (SELECT T1.U, AGG(Y:C)
FROM T1, Y
WHERE k2
Y.V operator2 R1.U
GROUPBY R1.U)
```

R1 and R2 are temporary relation where R1 is the set of all values in X.U that is employed to evaluate the inner block and R2 is the set of all aggregated values that is acquired in the inner block of Query AA. Each value is taken from the relevant value of X.U which produces it. R1 is acquired from the X-tuples that desire of k1 by projecting on X.U with duplicates removed and all membership degrees set to 1. R2 has got by joining R1 with Y-tuples that satisfy k2 on the join condition Y.V operator2 R1.U, grouping the result based on R1.U and use AGG to each of it.

Query AA can be used to unnested by make use of one of following two queries.

Query AA' is used when AGG is not CNT and Query CNT' is used, if otherwise.

```
Query AA'
SELECT X.A
FROM X, R2
WHERE k1 AND X.U≡ R2.U
AND X.B operator1 R2.A
Query CNT'
SELECT X.A
FROM X, R2
WHERE k1 AND X.U.+≡R2.U
[X.B operator1R2.A: X.B operator1 0]
```

The WHERE clause is a combination of the predicate k1 and a left outer join predicate which is followed by an IF-THEN-ELSE enclosed in a [ ]. The left outer join operator (Lacroix and Pirotte, 1976; Codd, 1979; Rosenthal and Reiner, 1984) is to preserve the tuples of the relation X because only X.A is projected.

EXPERIMENTAL RESULT

This unnested, nested fuzzy and horizontal aggregation queries are evaluated using java and SQL server as database. The dataset used in this study is employee dataset. This dataset contains employee salary, male, female employee and city details. In this section, we analyzed the data mining project by unnested queries. For this purpose, various unnested and nested queries are used to evaluate and analyze the dataset. The experiment conducted on Intel duo core as processor which has 8 GB hard disk, 4 GB ram and 2.40 GHz processor speed. The nested fuzzy and unnested query produces the same result. But nested fuzzy queries take more execution time and memory utilization than conventional unnested queries. The performance of these nested, unnested fuzzy queries and horizontal aggregation queries (CASE, SPJ and PIVOT queries) are analyzed by execution time, memory usage and cost of execution of queries.

Nested fuzzy queries, CASE, SPJ and PIVOT queries: In this section, execution time, memory usage, cost of nested fuzzy queries CASE, SPJ and PIVOT queries are discussed. Table 2 shows that the nested fuzzy queries, CASE, SPJ and PIVOT queries consumes high execution time, memory usage and cost. From Table 1, we clearly understand that each queries of nested fuzzy takes execution time in the range of 9.7

Table 2: Execution time, memory usage and cost of nested fuzzy queries CASE, SPJ and PIVOT queries

Query name	Execution time	Memory	Cost
Basic: type N	11.131862	61.6171875	61.0
Basic: type J	11.624662000000001	61.4033203125	61.0
Pivot query	11.545881	57.0947265625	57.0
Nested query with aggregate	9.7624139999999997	59.6884765625	59.0
Nested query set operator	11.180471000000001	60.955078125	60.0
CASE Query	11.489729000000001	60.2041015625	60.0
SPJ Query project join	10.226998	62.2275390625	62.0
SPJ Outer project join	10.64856	62.3798828125	62.0
Nested fuzzy query	17.779354999999999	61.2060546875	61.0

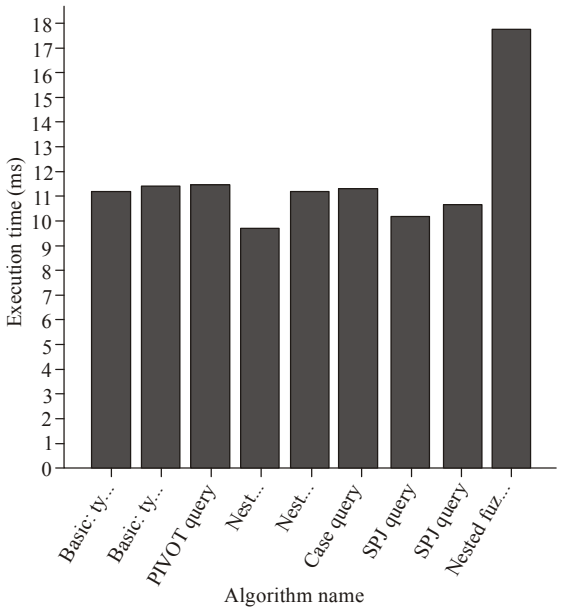

Fig. 1: Execution time of nested fuzzy queries, CASE, SPJ and PIVOT queries

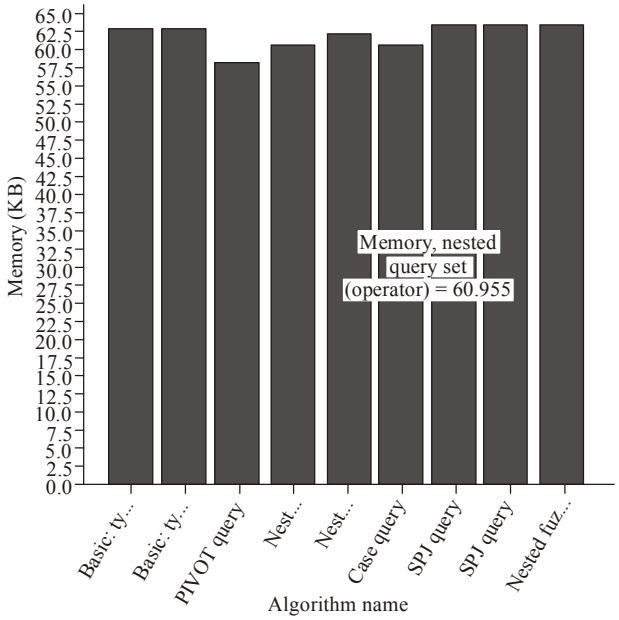

Fig. 2: Memory utilization of nested fuzzy queries, CASE, SPJ and PIVOT queries

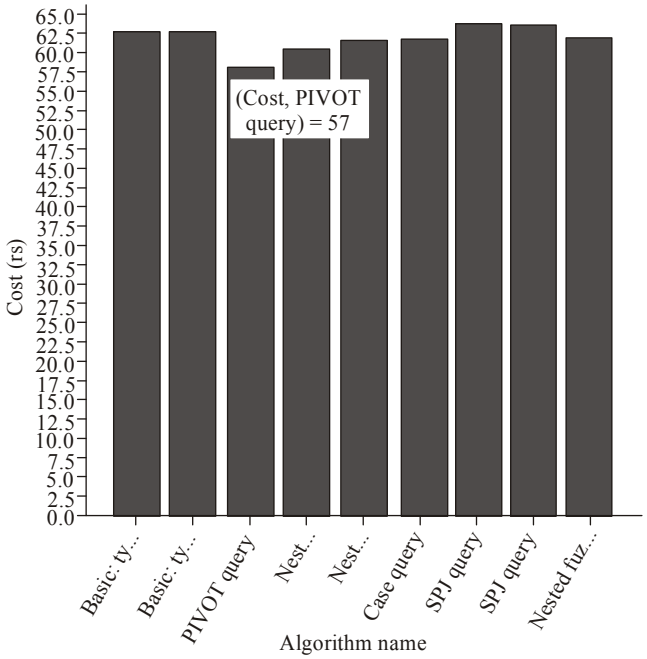

Fig. 3: Cost of execution of nested fuzzy queries, CASE, SPJ and PIVOT queries unnested queries

Table 3: Execution time, memory usage and cost of unnested fuzzy queries

Query name	Execution time	Memory	Cost
Unnested query	2.0407621250000001	53.897663247501349	53
Unnested query JA	7.8977913749999997	53.617389810001349	52
Unnested query count	9.1876202500000002	54.660358560001349	53
Unnested query Jx	1.3994446250000001	55.226764810001349	54

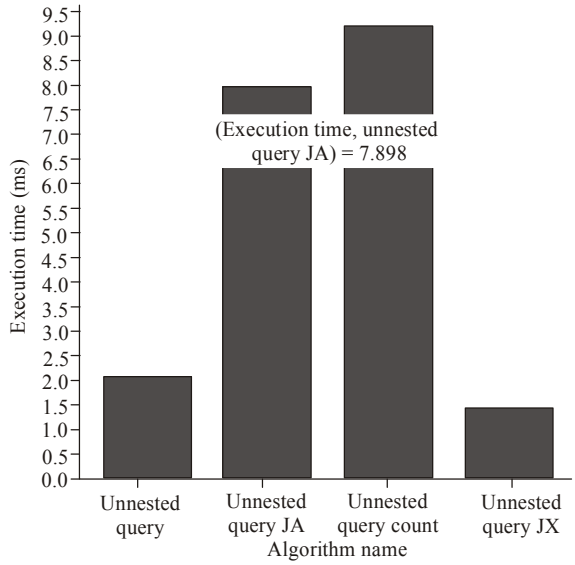

Fig. 4: Execution time of execution of nested fuzzy queries

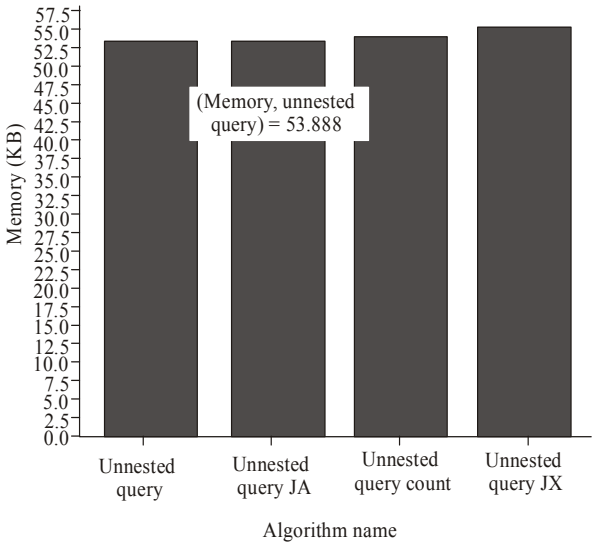

Fig. 5: Memory utilization of unnested fuzzy queries

to 17.7 ms, respectively memory utilization in the range of 57.09 to 62.37 kb, respectively and cost for execution of queries take 57 Rs to 62 Rs. Hence, the nested fuzzy queries take more time. The Pivot produces linear scalability than CASE and SPJ method (Ordonez and Chen, 2012).

Figure 1 to 3 shows Execution time, memory usage and cost of nested fuzzy queries, CASE, SPJ and PIVOT Queries. This execution time, memory utilization and cost are measured by milliseconds,

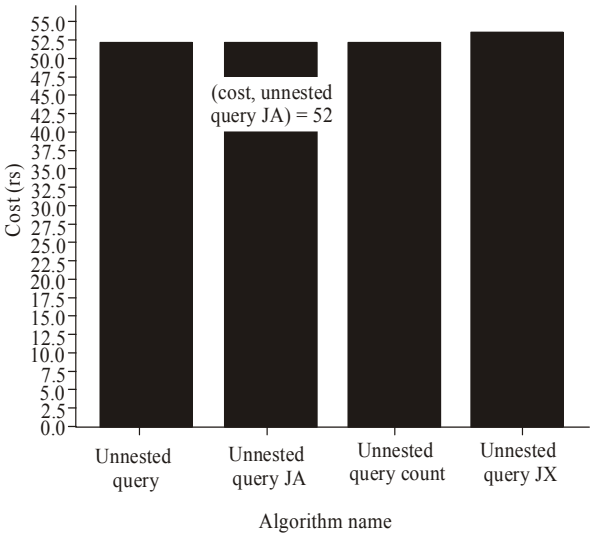

Fig. 6: Cost of execution of unnested fuzzy queries

kilobytes and amount. This figure is generated using bar chart in java.

Understand that each queries of unnested fuzzy take execution time in the range of 1.4 to 9.1 ms, memory utilization in the range of 53.6 to 55.2 kb, respectively and cost for execution of queries take 52 Rs to 64 Rs. Hence, the nested fuzzy queries take more time. Figure 4 to 6 shows execution time, memory usage and cost of unnested fuzzy queries. This execution time, memory utilization and cost are measured by milliseconds, kilobytes and amount. This figure is generated using bar chart in java.

This section discuss about execution time, memory usage and cost of unnested fuzzy queries. Table 3 shows that unnested fuzzy takes less execution time, memory usage and cost than nested queries. In Table 3, we proved that each unnested queries takes execution time in the range of 1.4 to 9.1 ms, respectively memory utilization in the range of 53.6 to 55.2 kb, respectively and cost for execution of queries take 52 Rs to 64 Rs. Hence, the unnested fuzzy queries take less execution time than nested queries because it uses join algorithm.

CONCLUSION

This study discussed about efficient processing of evaluation of Unnested and Nested Fuzzy SQL Queries for analyzing the data mining project. Performance of an extended merge-join with the unnested queries is compared with nested loop method in which the nested queries must be evaluated. In experimental results, we

have shown that the performance of evaluating the unnesting techniques with extended merge-join and horizontal aggregations techniques CASE, SPJ and PIVOT (Ordonez and Chen, 2012) in dataset. These techniques are applied to analyze the data mining in dataset.

REFERENCES

Baldwin, J.F., 1983. A fuzzy relational inference language for expert systems. Proceeding of the 13th IEEE International Symposium Multiple-valued Logic, pp: 416-423.

Codd, E.F., 1979. Extending the database relational model to capture more meaning. ACM T. Database Syst., 4(4): 397-434.

Gray, J., A. Bosworth, A. Layman and H. Pirahesh, 1996. Data cube: A relational aggregation operator generalizing group-by, cross-tab and subtotals. Proceeding of the 12th International Conference on Data Engineering (ICDE, 1996), pp: 152-159.

Kim, W., 1982. On optimizing an SQL-like nested query. ACM T. Database Syst., 7(3): 443-469.

Lacroix, M. and A. Pirotte, 1976. Generalized joins. SIGMOD Rec., 8(3).

Lohman, G.M., D. Daniels, L.M. Haas, R. Kistler and P.G. Selinger, 1984. Optimization of nested queries in a distributed relational database. Proceedings of the 10th International Conference on Very Large Data Bases (VLDB'84), pp: 403-415.

Ordonez, C., 2004. Vertical and horizontal percentage aggregations. Proceeding of the ACM SIGMOD International Conference on Management of Data (SIGMOD'04), pp: 866-871.

Ordonez, C., 2010. Statistical model computation with UDFs. IEEE T. Knowl. Data En., 22(12): 1752-1765.

Ordonez, C. and Z. Chen, 2012. Horizontal aggregations in SQL to prepare data sets for data mining analysis. IEEE T. Knowl. Data En., 24(4):678-691.

Qi, Y., Z. Weining, L. Chengwen, W. Jing, C. Yu *et al*., 2001. Efficient processing of nested fuzzy SQL queries in a fuzzy database. IEEE T. Knowl. Data En., 13(6).

Rosenthal, A. and D.S. Reiner, 1984. Extending the algebraic framework of query processing to handle outerjoins. Proceeding of the 10th International Conference on Very Large Data Bases (VLDB'84), pp: 334-343.

Wang, H., C. Zaniolo and C.R. Luo, 2003. ATLaS: A small but complete SQL extension for data mining and data streams. Proceeding of the 29th International Conference on Very Large Data Bases (VLDB'03), 29: 1113-1116.