Chapter 8

CONCLUSIONS

8.1 Summing-Up

In the earlier chapters the new mechanism of ONAM has been critically studied and offered as an effective substitute to the existing ineffective devices to provide rapid feedback to the users who issue nested queries with aggregates. The ONAM constitutes a framework for handling queries with multiple aggregates as well as an optimistic approach for dealing with the queries of multiple levels of nesting. This new mechanism evaluates a nested query progressively. Precisely, as soon as the inner query produces estimates to its aggregates, the outer query is evaluated to produce approximate answers to the users by taking even the least possible tuples into account. As the inner query’s aggregates are refined, the outer query’s aggregates are also refined too. The ONAM has been implemented using JAVA and MS-SQL Server, and the system itself has been evaluated. Results have shown that this mechanism is capable of providing rapid online feedback to the user without compromising on the quality of the answers.
In real-time systems an accurate answer if it is a second late is futile, compared to an approximate answer that is issued a second in advance. Thus ONAM's approximate answers in seconds are more effective thanaccurat answers in minutes/hours.

8.2 Contributions of ONAM

Thus the present study makes significant contributions in the area of Nested Aggregate Query Evaluation. These contributions are as follows:

(i) A new framework has been developed to evaluate the nested queries with multiple aggregates.

(ii) A new optimistic approach has been proposed for evaluating Aggregate queries with a single and multiple aggregates.

(iii) The optimistic approach has been extended to evaluate the multilevel nested aggregate queries.

(iv) For all the above three, the thesis contributes three new algorithms.

(v) The above three algorithms have been implemented.

(vi) Performance evaluation has been done for these algorithms.

The various aspects of the implementation, performance considerations and further work of ONAM with regard to a variety of parameters are given in the next subsection.
8.3 Future Research

The usability and performance needs of online aggregation model have not been elaborated in detail, so there seems to be much latitude in the solution space for the problem. This is beyond the scope of the present study. However, a few directions are listed below to facilitate further research in this field of study.

Since the Online Aggregation Model has been necessitated for better user interfaces, the processing of multilevel and multiple aggregates must be effectively shown in the GUI so that a user can understand the state of the query at any instant of time.

Since the accuracy of the estimation of the aggregation depends upon the statistical formulae used, computation of the running confidence intervals for various common aggregates presents a considerable challenge necessitating more work in this area of study.

One key issue to be explored is the level of statistical confidence that users demand during online aggregation. On one extreme, are the users who demand faultless accuracy; they will be satisfied only with complete answers, such as those returned by the batch system. On the other extreme, are the users who demand no accuracy at all; they will be satisfied by the wild guesses, and do not need a database system. The Online Aggregation Model provides a middle ground, quantitatively estimating the trustworthiness of the output.

Aggregation queries are generally long-running operations and take a much time. The long running operations must be checkpointed, so that computations can be saved in spite of system crashes, power failures and operators errors. The users should be allowed to continue the queries that they have earlier cancelled. Therefore
checkpointing the state of online aggregation queries proves useful.

The present work concentrates only on queries without joins, correlations, group-by queries. Not much light has been shed on these queries for obvious reasons. These queries can be taken up for more elaborate study, so that all their possibilities could be established.

This dissertation has dealt with only average aggregate function and, if necessary, the ONAM could be extended to other aggregate functions as well for academic interest.

NULL values are rather common in incomplete databases. Special care should be taken while dealing with such tuples, which come as a part of an aggregate query. In the ONAM, tuples, whose attribute values are NULL, are taken as ZERO.

Indexes play an important role in the online aggregation, which must be explored at greater length. Cluster index helps in accessing the tuples from the relation at a faster rate, facilitating rapid feedback.

Users are familiar with online display techniques from web-browsing software like Netscape Navigator, which display images that have been compressed using techniques such as interlacing and wavelets. These images can be very large, and are displayed as they are shipped over the network. As more and more of the image data arrives at the application, the user perceives the image getting as sharper and sharper. The display of a compressed image is essentially the output of an online aggregation over the bits of image. It may be interesting to try to unify the notions of online aggregation with interlacing wavelets and other compression techniques that present incrementally improving results.

ONAM can also be extended as a web-based implementation model.
It is clear that a good online aggregation system will require significant new research. However it seems equally clear that the relational systems should be extended to support online aggregation as well as traditional batch processing.

In order to generate correct running aggregates and estimates, tuples must be accessed in a random order from the relation. The precision of the running aggregates is meaningful only when the tuples are retrieved in a random order. This means that an online aggregation model should avoid access methods in which the attribute values of a tuple affect the order in which the tuple is retrieved, as elaborated below:

(i) Insertions

Whenever a new tuple is inserted in the relation, it should be placed in such a manner that the sequential scan of the relation gives meaningful running aggregates and estimates. As the DBMS is responsible for insertions and deletions, it should contain a mechanism by which the relation gets tuned and generates meaningful running aggregates and estimates even when the relation is sequentially scanned. Though costly during insertions, it generates a better performance over the conventional database by producing almost exact aggregates at the early stages of computation. This makes the user more comfortable with very good running aggregates and brings a new confidence on online aggregation, which can be utilized in effective decision-making.

(ii) Indexes

As the size of the relation increases the insertion becomes more complex and takes more time for insertion of tuples as mentioned earlier. An alternative to this is to access the tuples of the relation based on some other attribute, which facilitates to access tuples in a random manner for this column. Care should be taken that the two columns are not correlated.
(iii) Lists When there is no suitable index on any column to retrieve a particular column values at random, a list has to be maintained containing the order in which the tuples have to be accessed to generate meaningful current running aggregates and estimates. This also reduces the cost when compared with insertions. But it requires the maintenance of lists as an additional overhead to the database.

(iv) Adding a Column

An additional column is added to the relation containing randomly distributed values. This column is not a part of the original relation. Then cluster index is done based on the column, which helps in the retrieval of tuples from the original relation in a random order. This process gives better initial approximation. Though this method is little expensive in terms of memory, it gives better performance results even when the table is scanned sequentially.

All of the above-mentioned operations give additional performance when dealing with online aggregation when compared to conventional DBMS. To perform any of the above-mentioned operations effectively, the attribute on which the user performs a query frequently has to be identified. This identification can be done when the DBMS maintains the user profiles. A user profile contains the list of frequently issued queries by the user to the DBMS.

This dissertation on the effectiveness of the ONAM may be rounded off with reiteration of the thesis statement that the approximate answers within seconds are more purposeful and desirable than the accurate answers in minutes/hours in decision-support environments.
Appendix

$Z_{a/2}$ Values for Different Confidences

<table>
<thead>
<tr>
<th>Confidence</th>
<th>Confidence/2</th>
<th>$Z_{a/2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>90</td>
<td>0.45</td>
<td>1.645</td>
</tr>
<tr>
<td>91</td>
<td>0.455</td>
<td>1.69556</td>
</tr>
<tr>
<td>92</td>
<td>0.46</td>
<td>1.75111</td>
</tr>
<tr>
<td>93</td>
<td>0.465</td>
<td>1.81143</td>
</tr>
<tr>
<td>94</td>
<td>0.47</td>
<td>1.88143</td>
</tr>
<tr>
<td>95</td>
<td>0.475</td>
<td>1.96</td>
</tr>
<tr>
<td>96</td>
<td>0.48</td>
<td>2.054</td>
</tr>
<tr>
<td>97</td>
<td>0.485</td>
<td>2.17</td>
</tr>
<tr>
<td>98</td>
<td>0.49</td>
<td>2.32667</td>
</tr>
<tr>
<td>99</td>
<td>0.495</td>
<td>2.575</td>
</tr>
</tbody>
</table>