

# DSQoS - Distributed architecture providing QoS in Summary Warehouses

João Pedro Costa  
Dep. Informática e de Sistemas  
Instituto Superior de Engenharia de Coimbra  
Quinta da Nora, Apartado 10057  
3031- 601 Coimbra, Portugal  
+ 351 239 790 200  
jcosta@isec.pt

Pedro Furtado  
Dep. Engenharia Informática  
Universidade de Coimbra  
Pólo II - Pinhal de Marrocos  
3030 Coimbra, Portugal  
+ 351 239 790 000  
pnf@dei.uc.pt

## ABSTRACT

Data warehouses (DW) that store enormous quantities of data put a major challenge in what concerns performance and scalability, as users request instant answers to their queries. Traditional solutions rely on very expensive architectures and structures for speedup and scale-up. The Summary warehouse (SW) is an inexpensive solution that has the potential to deliver very fast approximate answers to aggregate queries using only general-purpose sampling summaries.

Although summaries are expected to be extremely fast, some analysis requires larger summaries to estimate individual group results, compromising the speedup advantage. This is the accuracy/speedup (A/S) tradeoff.

In this paper we propose the “Distributed Set-of-Summaries for Quality of Service” (DSQoS) that solves the A/S issue by optimizing the accuracy and response time for each query pattern in order to guarantee a desired Quality of Service (QoS). This QoS is defined in terms of response time and accuracy bounds. The strategy determines the required summary size to guarantee the accuracy targets and then dynamically select a set of summaries, distributed in various nodes, which can ensure the QoS constraints (time and accuracy). The strategy presents enormous possibilities since each node can contain summaries with different sizes, depending on the node characteristics, and can dynamically be added and removed from the system.

We discuss the design of the approach and the strategies used to process queries. In the experimental section we show how the approach is able to deliver almost instant and accurate answers without employing expensive architectures, which would be impossible using other strategies.

## Categories and Subject Descriptors

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.  
*Conference '00*, Month 1-2, 2000, City, State.  
Copyright 2000 ACM 1-58113-000-0/00/0000...\$5.00.

H.3.3 [Information Search and Retrieval]: Information Search and Retrieval - *retrieval models, search process*

H.3.3 [Information Search and Retrieval]: Systems and Software - *distributed systems, performance evaluation (efficiency and effectiveness)*

## General Terms

Algorithms, Management, Measurement, Performance, Design

## Keywords

Approximate Query Answering, DataWarehouse, Sampling, OLAP

## 1. Introduction

Data warehouses (DW) are becoming crucial tactical and strategic tools in profit-aware organizations and are increasingly used as a crucial day-to-day tool. In many important organizations, the applications that allow users to analyze data (e.g. iterative exploration) have to deal with Giga or Terabytes of data, while those users require almost instant answers.

This issue has prompted investments in expensive special hardware architectures and complex distributed and parallel computation systems that improve query response time in the presence of such massive data sets. However, as return-on-investment considerations become crucial in technology investments, organizations worldwide become less willing to invest large sums and time in special architectures and consulting when much cheaper and simpler solutions can be found.

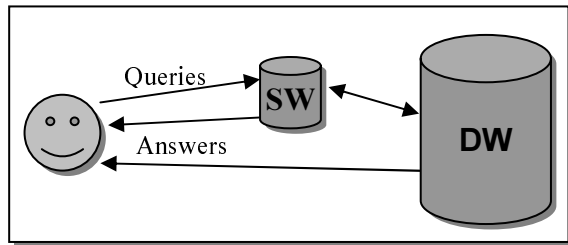
In the next section we discuss some of the most recent solutions and show that those solutions cannot guarantee or even deliver some sort of QoS. The discussion about the need of a QoS assurance is presented in section 3. Our architecture (DSQoS) that can deliver such QoS assurance is explained in detail in section 4 together with the analysis of the expected gains of the approach. Section 5 analyzes experimental results using the TPC-H decision support benchmark [12]. Section 6 contains concluding remarks.

## 2. Related work

Many strategies were proposed to overcome the performance problem aroused by the amount of data that RDBMS systems must handle to return answers to business queries.

In Data Warehouse Striping (DWS) [3] the authors propose distributing the whole data warehouse into nodes for performance purposes. DWS is a faster data warehouse architecture that can answer any question placed to the system, with an approximately linear speedup to the number of nodes (N). But for practical reasons the value of N is limited by the number of nodes. Each node of DWS computes his partial result that will be merged to obtain the answer to the query. Partial results can only be obtained if all dimensions of a star schema model are completely replicated into each node. This could significantly reduce the expected speedup when dimensions are large and their size is greater than the fact data in the node. DWS also includes the possibility of approximate answering to aggregated queries when nodes fail (DWS-AQA), however it takes as much time as answering with the full data, because available nodes must process the same amount of data.

There are also several recent works in approximate query answering strategies [1][4] [11][2][14]. Hellerstein [11] proposed a framework for approximate answers of aggregation queries called online aggregation, in which the base data is scanned in random order at query time and the approximate answer is continuously updated as the scan proceeds. A graphical display depicts the answer and a confidence interval as the scan proceeds, so that the user may stop the process at any time. The Approximate Query Answering (AQUA) system [7][8] provides approximate answers using small, pre-computed synopsis of the underlying base data. The system provides probabilistic error/confidence bounds on the answer [2][9].



**Figure 1 – Sampling summary strategy**

Sampling summaries can return a huge speedup because they are small data sets that can answer any aggregation query pattern as long as they have enough “representation capacity”.

The accuracy limitation of summaries comes from the fact that they must estimate results for individual groups independently. As the user drills down to individual product or week, the estimation may become impossible. The lack of samples in summaries is considered an important issue and there are some previous works on the issue [1][2][4]. However, these solutions have limitations because they are pattern-oriented.

A possible solution to this problem can be achieved by constructing a larger summary so that it can answer most query patterns, however it will be large and slow. This is important when typical data warehouses contain hundreds of Gigabytes or Terabytes of data and even small percentage summaries are still very large for queries that demand fast answers (e.g. a 5% summary is still very large).

The DS approach [5] was conceived to overcome the speedup limitations on approximate answering. By distributing a summary

into several nodes, we can obtain a great speedup and at the same time allow the enlargement of the summary size so that more detailed queries can be answered and with improved accuracy. This strategy does not specify how one can determine the adequate summary given accuracy and speedup targets.

In [6] we proposed the use of a Bag-of-summaries (BofS) which is a set of summaries with different sizes to handle different query requirements. However, BofS cannot provide response time guarantees because less aggregated queries require larger and slower summaries.

The DSQoS strategy proposed in this paper adds the necessary summary choice heuristics to distributed summaries to guarantee response time and accuracy targets.

### 3. Why do summaries need QoS?

When summaries are chosen to improve performance of aggregation queries, it is important to know or at least define some quality of service (QoS). We cannot just construct a summary and hope that it will solve our problems, because it could be too large or too small. If the summary is large enough to return results with acceptable accuracy, it may be too large to return an answer within acceptable time. On the other hand, if it is small enough to return results within an acceptable time, the accuracy may be unacceptable.

The suitable summary size that can return acceptable accuracy within acceptable response time depends on several factors, such as the query aggregation granularity (more detailed queries demand larger summaries), the desired accuracy (expressed with confidence intervals) and the admissible response time. The aggregation granularity determines the number of groups in which the sample set must be divided and so influences the number of samples that exist in each aggregation group. Figure 2 shows a typical TPC-H aggregation query.

```
SELECT  p_brand, Year_Month, l_discount,
        avg(l_quantity), sum(l_quantity), count(*)
FROM    lineitem, part
WHERE   l_partkey=p_partkey
GROUP BY p_brand, Year_Month, l_discount
```

**Figure 2 – Typical Aggregation Query (TPC-H) – Query Qa**

The resulting accuracy depends, among other aspects, on the number of samples of each aggregation group. Figure 3 shows the confidence intervals (C.I.) for the AVERAGE, COUNT and SUM, using the Central Limit Theorem (CLT), where  $z_p$  is the confidence level,  $\sigma$  is the standard deviation,  $n_s$  is the number of samples,  $N$  is the population size and  $ls$  is the sum of samples (linear sum).

<b>AVG</b>	$Z_p \frac{\sigma}{\sqrt{n_s}}$
<b>COUNT</b>	$Z_p \frac{\sqrt{n_s}}{SP} = Z_p \frac{N}{\sqrt{n_s}}$
<b>SUM</b>	$Z_p \frac{\sigma}{SP} (\sqrt{n_s} + Z_p) + \frac{ls}{n_s} Z_p \frac{\sigma}{\sqrt{n_s}}$

**Figure 3 – L.C.T Confidence intervals (C.I.)**

Intuitively or from the analysis of the formulas it is possible to conclude that, in order to have accurate estimations, it is important to have at least a reasonable number of samples. Going back to the query in figure 2, the estimation procedure is applied individually within each group aggregated in the query. In that case the query was aggregating by brand and month, so the estimations must be produced within each combination brand/month. In exploration analysis, the user might then drill-down to product or week, or something completely different. In practice this means that aggregation granularities (degrees of detail) vary widely, so that there may not be enough samples for some estimation. We are limited by the representation capacity of a summary with a given size. Even if the summary can estimate the sales of some brands by week, groups are heterogeneous in size (and data distribution as well), so that it may lack samples for others.

This mean that greater summary sizes are needed when aggregation queries are more detailed and when the desired accuracy increases. So it's important to have not one but a set of summaries that are used according to the query aggregation and desired accuracy. To be easily usable and transparent to the user it is then important to have some sort of mechanism that can dynamically choose the most adequate summary to answer each query so that more detailed queries could be executed more rapidly with smaller summaries. This issue was first explored on BofS[6].

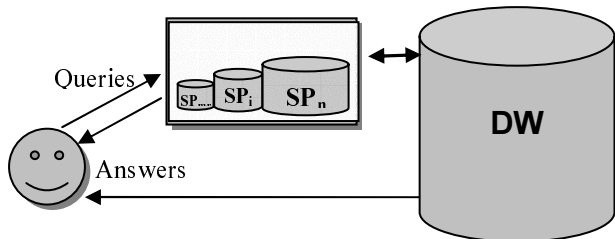


Figure 4 – BofS Architecture

By choosing the best summary to answer a query, BofS returns more speedup for more aggregated query patterns. However, it does not give response time guarantees. The user must pay attention to the degree of the aggregation detail when exploring the data, or else, he can wait much longer than expected (or desired) and in some cases the system could be accessing the whole DW when there is no summary large enough to answer the query.

#### 4. DSQoS Strategy

As discussed before, it is important to be able to increase the summary sizes without exceeding the desired response time. This could be achieved using DSQoS strategy. In this section we explain DSQoS architecture and how it works. DSQoS is based on determining the most appropriate summaries to be placed in computing nodes to provide accuracy and acceptable execution time and providing summary choice heuristics and processing strategies to handle queries using those summaries. We explain how the strategy determines the summary sizes that should be used, distributes the summaries and processes the queries. Finally, we analyze and compare the expected speedup of DSQoS.

#### 4.1 DSQoS Architecture

To achieve the desired goal, DSQoS uses distributed computing with several nodes. Instead of having just one node with a set of summaries that could be chosen dynamically according to the query granularity, we could have several nodes containing a set of summaries. The nodes could be the computer of users that explore the DW data or any other available computer. Each machine that participates in the architecture will manage a set of small summaries, with huge speedups, that when combined together with summaries existing in other nodes form a summary large enough to answer every aggregate query with the desired accuracy.

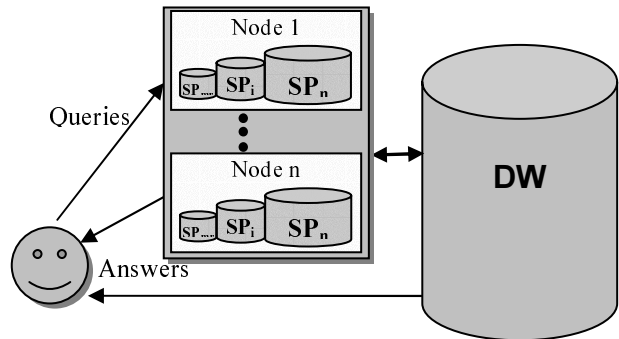


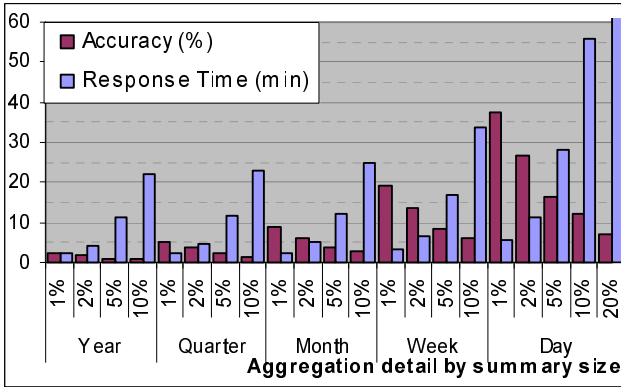
Figure 5 –Structure of the DSQoS Architecture

Figure 5 shows a DSQoS configuration with two identical nodes where each node has three summaries that can answer within different response time targets. Summaries from different nodes can be used in parallel followed by results merging to provide either faster or more detailed analysis of data (requiring larger SP than available in one node).

#### 4.2 How does DSQoS deliver QoS?

**Accuracy QoS:** the desired accuracy can be guaranteed by building a set of summaries (or “bag-of-summaries“(BofS)) in each node with various sizes that are proper to answer different aggregation granularities. The system chooses the best combination of summaries to answer queries within accuracy/response time targets. Figure 6 shows the response time and accuracy for different aggregation details and summary sizes. More detailed queries must use bigger summaries (with poor response time) while less detailed queries could be answered with the smallest summary that guarantees the desired accuracy. For instance, to obtain a 10% accuracy guarantee, the Qa yearly, quarterly and monthly aggregation can be answered by the 1% or 2% summary, the weekly aggregation requires a 5% summary and the daily aggregation requires a 20% summary. The 20% summary exhibits a response time that may be unacceptable for the user.

**Response time QoS:** for response time QoS, the user specifies that those accurate answers be returned within the specified time constraints. For instance, the user may also specify that the response time take more than 1min but less than 5min. In this case the Qa yearly, quarterly and monthly aggregation could be answered by the 2% summary but the weekly and daily aggregation couldn't be answered because the time guarantees are not meet.



**Figure 6 – Response time and accuracy for different aggregation and summary size of the query Qa**

The distributed computing takes a key role in the DSQoS architecture in order to guarantee QoS. DSQoS incorporates BofS logic to optimize query response time, but also determines which summaries to construct in a network of parallel computing nodes to guarantee desired response times. As a result, it can deliver desired scale-up and speedup regardless of the aggregation detail, as long as there are concurrent nodes available.

### 4.3 DSQoS Management

DSQoS is an inexpensive solution, as no special hardware is required, and it is very simple to manage, as the configuration of nodes is flexible and completely dynamic: nodes can enter or leave the system, be online or offline at any given time. The distributed organization allows any candidate node in a local network to host summaries to collaborate in the process of computing query results. When a node enters DSQoS, a star schema similar to the base data schema is created for each summary and loaded with samples from the DW facts. The corresponding dimension values are also loaded according to the sampled summary facts. Two key aspects need to be defined: the dimension of each of the summary and the amount of summaries. To determine the summary size is important to take into accounting the response time and accuracy requirements.

Accuracy targets are expressed as  $CI_{target} = CI\% = CI_{estimation} / estimation$ , the relative CI (e.g. the error should be within 10% of the estimated value). Response time targets can be expressed by means of a parameter “maximum response time”(MRT). This means that a query answered within MRT is tolerable, but the user does not wish answers to take more than MRT. Figure 7 shows the QoS parameters and their objectives.

DSQoS has an additional parameter - mRT (minimum response time) – below which (MRT-bound for accuracy) is used and above which (CI-bound for speed) is preferred. This parameter mRT is useful to determine the size of the smallest summary that is dimensioned to answer a query workload in time mRT.

Parameter	Objective	Values
CONFIDENCE (x%) <b>ci%</b>	Enable / Inhibit approximate answers	x% =75 - 100% ci% = 5 - 100%
MAX RESPONSE TIME <b>MRT</b>	Set maximum response time	Typical <b>MRT</b> 30 secs to 20min
MIN RESPONSE TIME <b>mrt</b>	Set minimum response time	Typical <b>mrt</b> 1 secs to 30 secs

**Figure 7 – QoS parameters**

The best summary choice must take into account two different perspectives:

- It must answers as fast as possible within the CI accuracy target (CI-bound for speed);
- It must be as large as possible (best accuracy) within MRT (MRT-bound for accuracy)

DSQoS uses a typical aggregation query workload QW (a set of queries posed historically or some benchmark queries) to determine the largest summary SMRT that would be capable of answering within MRT time guarantees and the smallest summary SmRT that answers at or below mRT. The sampling rate of SMRT (SPMRT) can be determined by a simple heuristic procedure based on the fact that if the summary processes only SP of the data, then it must be at least 1/SP faster. We included one or more fine-tuning steps because the actual summary speedup is typically  $x/SP \approx 1/SP$ , where x is a relevant multiplicative factor. The algorithm is:

**First try:**  
 Run QW once against the data warehouse to obtain  $t_{dw}$   
 (e.g.  $t_{dw}$  such that 10% responses slower,  
 or  $t_{dw}$  is slowest response)  
 Set  $SP_{MRT} = MRT / t_{dw}$  and obtain  $S_{MRT}$  by sampling  
 (one-pass sequential sampling [Vitter])

**Fine-tune:**  
 While [  $t(S_{MRT})$  not within  $MRT \pm \delta$  ] ( $\delta$  is a tolerance,  
 e.g. to be within 10% of MRT)  
 Run QW against  $S_{MRT}$   
 Increase (/decrease) SPMRT by  $MRT / t(S_{MRT})$   
 by adding(/taking) samples

The SMRT summary size is only indicative, as it was determined based in a query workload QW. To have an SMRT that typically guarantees the MRT response time, the query workload QW should be tilted towards complex, slow queries. The minimum summary (SmRT) is computed similarly from either  $t_{dw}$  or from  $t(S_{MRT})$ .

Summaries should be chosen or determined automatically to cover the spectrum between these two limits [SmRT, SMRT]. The set of summaries in a node is therefore (SmRT, ..., SMRT),  $SmRT < \dots < SMRT$  (SMRT could be decreased by a small value  $\Delta$  to account for the overhead of data exchange between nodes and partial result merging from those nodes, but this value is typically negligible).

The system places summaries with sizes between SPmRT and SPMRT in each computing node of a network, and/or uses the existent ones. When queries are posed to, the system rewrite and redirect them in parallel and independently using the nodes, so that it is able to meet the MRT requirement (approximately) and return accurate results simultaneously.

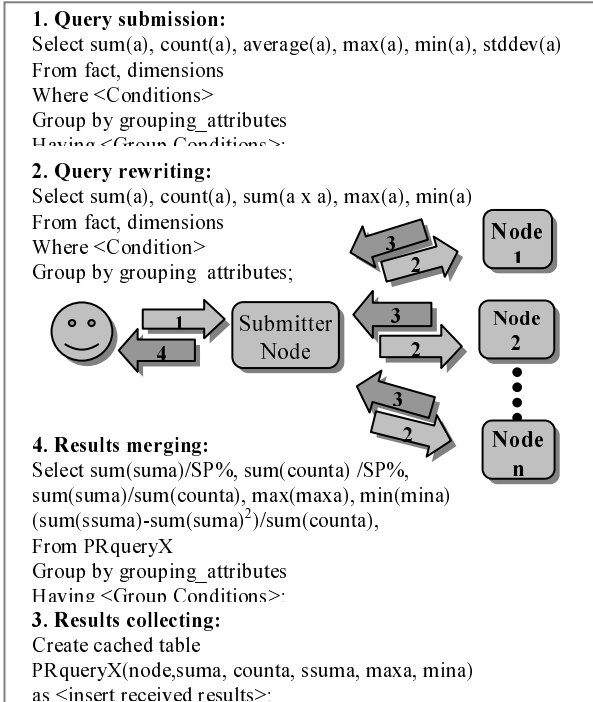
The network can be a simple LAN and nodes can be ordinary machines that serve other purposes as well. Nodes can enter or leave the system dynamically and the system is designed so that answers are obtained regardless of the number of nodes that are online at a given moment. Each DSQoS node j is loaded with at least the summary  $S_j = (SMRT(j))$  and preferably also with the

remaining, smaller summaries. In our design the samples from the summary  $\sum \text{SMRT}(i)$  should be obtained through random sampling without replacement (e.g. using one pass reservoir sampling [13]).

#### 4.4 DSQoS Query Processing Strategy

Queries are parsed and either submitted against DSQoS if they are aggregations, or otherwise redirected into the original DW. The aggregation query Q is evaluated to determine the required summary size SPQ needed to answer the query within the defined targets. Based on SPQ, there are two alternatives for query Q:

- **Local processing** (e.g. when offline): The query is rewritten to run locally using the smallest summary S such that  $\text{SPS} > \text{SPQ}$ . If  $\text{SPQ} > \text{SPMRT}$  the query cannot be answered locally;
- **Distributed processing** (online): the query is processed in parallel against summaries  $\text{SP}(i)$  from more than one node and then merged. If n nodes are available, the fastest query response time can be obtained by choosing the smallest  $\text{SP}(i)$  such that  $\sum \text{SP}(i) > \text{SPQ}$ . Distributed computing of summary answers requires query rewriting and distribution by the submitting node, independent execution by available nodes, collection and merging by the submitting node to obtain the final results, as showed in figure 8.



**Figure 8 – DSQoS Query Processing**

DSQoS uses a single virtual summary composed with samples of all component summaries and with a global summary sampling percentage SP obtained as the sum of the sampling percentages of the component summaries ( $\text{SP} = \text{SP}(1) + \dots + \text{SP}(n)$ ). However, the set of samples is not available in the merging node and the objective is to make each node compute a partial answer independently, with the controlling node merging the independent partial answers. This is obtained by computing additive parameters in individual nodes and using the following formulas to arrive at the final estimation:

$COUNT_{estimated} = \frac{COUNT_{sample\_set}}{SP\%} = \sum_{all\_nodes} \frac{COUNT_{sample\_set\_node_i}}{SP\%}$
$SUM_{estimated} = \frac{SUM_{sample\_set}}{SP\%} = \sum_{all\_nodes} \frac{SUM_{sample\_set\_node_i}}{SP\%}$
$AVERAGE_{estimated} = \frac{\sum_{all\_nodes} SUM_{sample\_set\_node_i}}{\sum_{all\_nodes} COUNT_{sample\_set\_node_i}}$
$STDDEV_{estimated} = \sqrt{\frac{(\sum SS_{sample\_set\_node_i} - \sum SUM_{sample\_set\_node_i}^2)}{\sum COUNT_{sample\_set\_node_i}}}$ <p style="text-align: center; margin-top: -10px;"><i>SS (SUM OF SQUARES)</i></p>
$MAX_{estimated} = MAX(MAX_{sample\_set\_node_i})$
$MIN_{estimated} = MIN(MIN_{sample\_set\_node_i})$

This means that the query rewriting step needs to replace each AVERAGE and STDDEV (or variance) expression in the SQL query by a SUM and a COUNT in the first case and by a SUM, a COUNT and a SUM\_OF\_SQUARES in the second case, while it can keep the other operators as they are. For typical aggregation queries, those are the most relevant modifications, as the GROUP BY clause is left as-is and the HAVING clause (if present) can be eliminated from the rewritten query for the different nodes and only applied later in the result merging phase.

The query submitter receives partial results and merges them. Simultaneously, it computes confidence intervals and publishes the results. In practice, this means running a modified version of the initial query against a temporary cached table that contains the partial results returned by the nodes. Figures 8 shows exactly what each step does using an example.

Confidence intervals (CI) are also computed directly in step 4 using the formulas described in section 3. The results can be published “on-the-fly”. If the node is offline, the results are published immediately. Otherwise, the system can start delivering results as soon as there are any and as additional results arrive, the system merges and posts the answer to show improved results.

#### 4.5 Expected Speedup of DSQoS

Consider a network with N similar nodes running DWS: DWS(N). Consider also a summary with sampling percentage SP: SW(SP). Finally, consider DSQoS with N nodes: DSQoS(N,SP1,...,SPn) (for simplicity we consider equal summary sizes in all nodes and neglect the data exchange and merge overheads).

The expected speedup of a one-node summary SW(SP) answering a query is  $x/SP > 1/SP$  (the summary has only SP% of the DW data). The expected speedup of DWS(N) is about N [3] (approximately linear with the number of nodes). DSQoS obtains an expected speedup of  $Nx / [\min(\text{SP}_i); \sum \text{SP}_i > \text{SPQ}]$  (the

minimum size chosen within each node that guarantees the condition  $\sum SP_i > SP_Q$ .

Configuration	Expected SpeedUp
DW	1
DWS	N
SW	$X/SP \approx 1/SP$
DSQoS	$NX / [\min(SP_i); \sum SP_i > SP_Q]$ $\approx N / [\min(SP_i); \sum SP_i > SP_Q]$

The following table shows the response time for a query taking 10 hours against the base data (DW).

Configuration	Expected SpeedUp
DW	10 hours
DWS	1 hour
SW	6 minutes (if SP <sub>q</sub> =1%)      30 minutes (if SP <sub>q</sub> =5%)
DSQoS	0.1x10 = 20 seconds      0.5x10 = 72 seconds

Not only DSQoS obtains extremely fast answers, as it overcomes the limitation of individual summaries (SW) that have a minimum granularity that they can answer. DSQoS can answer very detailed exploration analysis very accurately and still much faster than any other alternative. For instance, DSQoS could answer even a query requiring SP<sub>Q</sub> ≥ 25% using 2.5% x 10, with a speedup of 400 times (less than 15 minutes).

## 5. Experimental Analysis

This section analyses experimental results. The evaluation was conducted on a 5 node network with Intel PentiumIII 866MHz processor, 120GB IDE hard disk and 512MB RAM, running Windows 2000 professional with Oracle 8i DBMS, running TPC-H decision support benchmark™[12] with scale factor (SF) 100(100GB). Section 5.1 shows how DSQoS determine the size of the summaries for a given MRT and mRT choices and a query workload. Section 5.2 analyzes the response time, error and accuracy returned by DSQoS to query Qa (figure 2), where we can see that the predicted query response time and confidence intervals are attained.

### 5.1 Setting up DSQoS

Figure 9 shows how DSQoS summaries were determined from a given MRT and mRT and a query workload involving a query Qa shown in figure. Query Qa is a slightly modified version of a typical TPC-H query, to test varied granularities (degrees of detail). Qa computes aggregate quantities per brand, discount and a defined time period (e.g. year, quarter, month or week). Qa(Weekly) was used in the workload QW. The figure also shows the intermediate computations involved in determining SPMRT and SPMRT.

Parameter	Values	Results
QW(SF100)	Qa(weekly), QBrand	Choose slowest (Qa(weekly)) → t <sub>dw</sub> = 107.7 mins

	(yearly)	
MAX RESPONSE TIME MRT	6 mins	<b>First try:</b> $SP_{MRT} = 6 / 107.7 = 5.6 \%$ → resp. time = 11.09 min <b>Fine-tune:</b> subtract 1 - 6/11.09 = 46.2% samples New $SP_{MRT} = 5.60\% \times 0.53 = 3.01\%$ → resp. time = 5.90 min within 10%
MIN RESPONSE TIME mrt	10 secs	<b>First try:</b> $SP_{mrt} = 10 / (5.9 * 60) * 3.01\% = 0.1\%$

Figure 9 – calculation of the sizes of DSQoS summaries

### 5.2 Running Queries on DSQoS

Figure 10 shows the computation of the minimum number of samples required for the AVERAGE and COUNT aggregation functions considering CI of 5%, 10% and 15%. It also shows the characteristic number of elements per aggregation group (ng). This number can be determined using some selectivity estimation strategy. In this case the weekly value was collected from the execution of the query workload (QW).

Min Samples	Average	Count
Formula	$(z_p / ICrel)^2 (\sigma / \mu)^2$	$(z_p / ICrel)^2$
ICrel = 5%	$(1.65 / 0.05)^2 * 0.25 = 273$	$(1.65 / 0.05)^2 = 1089$
ICrel = 10%	$(1.65 / 0.10)^2 * 0.25 = 68$	$(1.65 / 0.10)^2 = 273$
ICrel = 15%	$(1.65 / 0.15)^2 * 0.25 = 31$	$(1.65 / 0.15)^2 = 121$

Figure 10 – Determination of the minimum number of Samples

Figure 11(a, b, c) shows the summaries chosen by DSQoS to execute query Qa with various granularities (Weekly, Monthly, Quarter and Year) and different accuracy targets (5%, 10% and 15%) in the 5 node system with S = (0.1%, 0.25%, 1%, 3%) (these summaries were determined above). The response times include data exchange and merging overheads.

We provide an example to help analysing the results shown. For instance, the monthly aggregation with CI=10% was answered in 34.4 seconds, using the second smallest summary (0.25% x 5 nodes). The CI target (10%) was perfectly met (CI<sub>Q</sub>=5.15%) and the error was small as well (EQ=2.34%). If the node were offline, the query would still be answered by the 1.00% summary in 1.75 minutes. The same query posed against the original DW took 102 minutes and against the DWS would take about 19.9 minutes (to take 1 minute the DWS would need about 100 nodes, and we are not accounting for the data exchange and merge overheads).

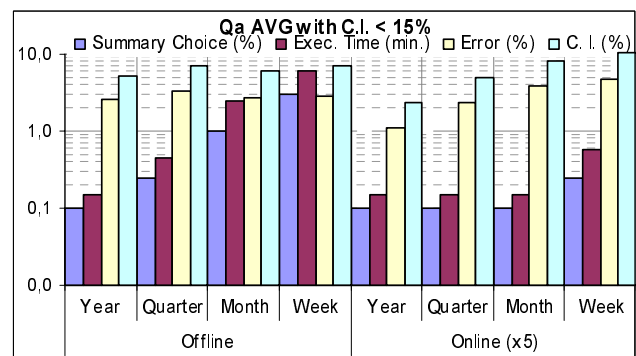


Figure 11-a – Response Time and Error Qa (C.I < 10%)

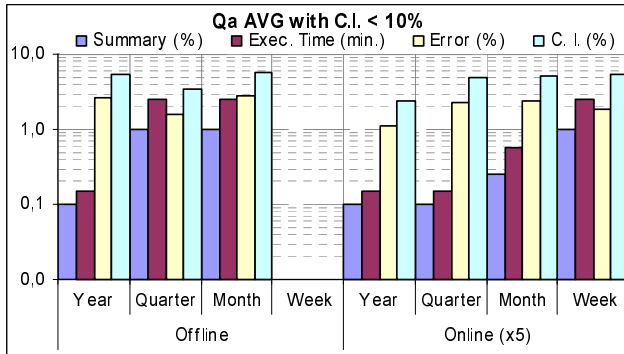


Figure 12-b – Response Time and Error Qa (C.I < 10%)

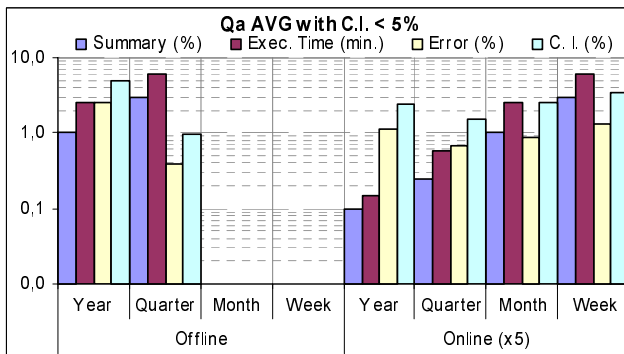


Figure 13-c – Response Time and Error Qa (C.I < 10%)

The results are very satisfactory, as queries were executed as fast as possible and the average error (EQ) and CI (CIQ) returned well within the target CI (CIrel). We can also see that more aggregated queries answer faster than less aggregated ones, as expected. For instance, for the 10% CI target, the weekly aggregation took 2.5 minutes (1 % summary x 5), while the month aggregation took 34.4 seconds (0.25% summary x 5) and the year aggregation took only 7.9 seconds (0.1% summary x 5).

The node data exchange and merging overheads of DSQoS are typically small. The next table shows the time-consuming overheads of DWS for query Qa (yearly and quarterly aggregation) in a 10 nodes network which can be used as a majoring value of DSQoS overhead. The results show that, even though the number of group results returned by the query is relatively large, the total extra overhead is conveniently small.

Overhead component	Qa Yearly	Qa Quarterly
Send and collect data from all nodes	1.04 secs	1.08 secs
Merge partial results	1.09 secs	2.07 secs
<b>Total</b>	<b>2.13 secs</b>	<b>3.15 secs</b>

**Experiment conclusions:** from these results we see that DSQoS can achieve extremely fast answers very cheaply for large data warehouses and provide response time guarantees. In fact, DSQoS is so flexible that it is possible to tune a system with a number of nodes and summary sizes to provide almost instant answers, even if the DW is extremely large.

## 6. Conclusions

In this paper we have proposed the use of “Distributed Set of Summaries providing Quality of Service” (DSQoS) to provide extremely fast approximate answers with response time and accuracy guarantee to aggregation queries. The approach is based on determining an appropriate set of summaries and placing them into a network of computing nodes based on response time targets for a query workload. The system can choose the best summaries from the available nodes to process the query in parallel. We have provided both analytical and experimental evidence showing the importance of the approach which is dynamic (nodes can enter or leave), flexible, cheap and simple to manage.

## 7. References

- [1] S. Acharaya, P.B. Gibbons, and V. Poosala. “Congressional Samples for Approximate Answering of Group-By Queries”, ACM SIGMOD Int. Conference on Management of Data, pp.487-498, June 2000
- [2] S. Acharaya, P.B. Gibbons, V. Poosala, and S. Ramaswamy. “Join synopses for approximate query answering”, ACM SIGMOD Int. Conference on Management of Data, pp.275-286, June 1999
- [3] Jorge Bernardino, Pedro Furtado, Henrique Madeira: “DWS-AQA: A Cost Effective Approach for Very Large Data Warehouses”. IDEAS 2002: 233-242
- [4] Pedro Furtado, João Pedro Costa: “Time-Interval Sampling for Improved Estimations in Data Warehouses”. DaWaK 2002: 327-338
- [5] Pedro Furtado, João Pedro Costa: “The BofS Solution to Limitations of Approximate Summaries” DASFAA 2003
- [6] Pedro Furtado, João Pedro Costa: “Distributed Summaries: Fast Accurate Summary Warehouses” IDEAS 2003
- [7] P. B. Gibbons and Y. Matias. “New sampling-based summary statistics for improving approximate query answers”. In Proc. ACM SIGMOD Int. Conference on Management of Data, pp.331-342, June 1998
- [8] P. B. Gibbons and Y. Matias. “AQUA: System and Techniques for Approximate Query Answering”. Bell Labs TR 1998
- [9] P. J. Haas. “Large-sample and deterministic confidence intervals for online aggregation”. In Proc. 9th Int. Conference on Scientific and Statistical Database Management, August 1997
- [10] J.M. Hellerstein, P.J. Haas, and H.J. Wang. “Online aggregation”, ACM SIGMOD Int. Conference on Management of Data, pp.171-182, May 1997
- [11] TPC Benchmark H, Transaction Processing Council, June 1999. Available at <http://www.tpc.org/>
- [12] J. S. Vitter. Random sampling with a reservoir. ACM Transactions on Mathematical Software, 11(1):37-57, 1985
- [13] J. S. Vitter and M. Wang. “Approximate computation of multidimensional aggregates of sparse data using wavelets”, ACM SIGMOD Int. Conference on Management of Data, pp.193-204, June 1999

