DWS-AQA: A Cost Effective Approach for Very Large Data Warehouses

Jorge Bernardino
Institute Polytechnic of Coimbra
ISEC - DEIS, Portugal
jorge@isec.pt

Pedro Furtado, Henrique Madeira
University of Coimbra
DEI, Portugal
pnf, henrique@dei.uc.pt

Abstract

Data warehousing applications typically involve massive amounts of data that push database management technology to the limit. A scalable architecture is crucial, not only to handle very large amount of data but also to assure interactive response time to the users. Large data warehouses require a very expensive setup, typically based on high-end servers or high-performance clusters. In this paper we propose and evaluate a simple but very effective method to implement a data warehouse using the computers and workstations typically available in large organizations. The proposed approach is called data warehouse striping with approximate query answering (DWS-AQA). The goal is to use the processing and disk capacity normally available in large workstation networks to implement a data warehouse with a very reduced infrastructure cost. As the data warehouse shares computers that are also being used for other purposes, most of the times only a fraction of the computers will be able to execute the partial queries in time. However, as we show in the paper, the approximated answers estimated from partial results have a very small error for most of the plausible scenarios. Moreover, as the data warehouse facts are partitioned in a strict uniform way, it is possible to calculate tight confidence intervals for the approximated answers, providing the user with a measure of the accuracy of the query results. A set of experiments on the TPC-H benchmark database is presented to show the accuracy of DWS-AQA for a large number of scenarios.

1. Introduction

During the last decade, data warehouses have become increasingly important in many applications areas such as decision support, banking and financial services. However, one of the aspects that most restrict the introduction of data warehouses in the organizations is the large investment in powerful servers needed to support large amount of data and to assure the interactive response time necessary to the *ad-hoc* querying typically used in decision support activities.

With the advent of fast networking technologies, many organizations have today large networks of computers (typically, personal computers and workstations) in which only a small fraction of the processing power and disk capacity are normally used. The parallel processing community has been seduced by the huge processing power available in many networks for quite a long time. However, this potentially available processing power has proved to be very hard to be used, even for the most favorable parallel applications.

We decided to investigate the possibilities of using the processing power and disk capacity normally available in large networks to implement a data warehouse at very low infrastructure cost. To achieve this we propose a new method called DWS-AQA (Data Warehouse Striping with Approximate Query Answering) based on the clever combination of two simple ideas: 1) uniform data striping to partition the data warehouse facts over an arbitrary number of computers, in such a way that queries can be executed in a true parallel fashion (a query is actually split into many partial queries), and 2) an approximate query answering strategy to deal with the momentary unavailability of one or more computers in the network.

The data striping technique is in fact the well-known round-robin partitioning applied to the very low level of the data warehouse facts in such a way that assures uniform distribution of the partitioned data. This very simple and old partitioning technique proved to be very effective for the typical queries in data warehouses. In fact, using this partitioning technique we can convert one query into queries that compute partial results, and the global result can be computed very fast from these partial results. The (partial) queries are executed independently in the different computers, which is the necessary condition to achieve optimal load balance and performance scaleup. This partitioning technique, called "Data Warehouse Striping" (DWS), was thoroughly evaluated in a previous work [5].

The key aspect that makes it possible to use the available processing power and disk capacity in a network

to implement a data warehouse is the combination of the DWS technique mentioned above with an approximated query answering scheme. Tight confidence intervals for the approximated answers are calculated, providing the user with a measure of the accuracy of the query results.

The main contributions of this paper are as follows:

- Proposes a framework to implement a large data warehouse in an affordable way. This approach has a very good scalability, as more computers can be added to the system as needed, and a nearly linear speedup can be obtained.
- Proposes methods that allow the computation of the different types of aggregation queries on an environment formed by a large number of computers. An efficient way to distribute the queries by all computers and merging the results is also presented.
- An exploration interface is proposed providing continuously feedback to the user and offering accurate estimates of the results, together with confidence intervals. This interface receives the partial answers from different nodes asynchronously, taking into account the unavailability of DWS-AQA.
- Presents an extensive set of experiments to establish the accuracy of DWS-AQA and provide statistical bounds for the errors contained in the estimations of final results.

The remainder of the paper is organized as follows. In the next section we review related work and discuss the problems associated with parallel and distributed data warehouses. Section 3 describes the different components of a DWS-AQA system. We present an experimental evaluation in Section 4 and the final section summarizes the conclusions from this work.

2. Related work

Today's commercial database systems support creation and use of indexes and materialized views. Materialized views are based on pre-computed answers to queries and are probably the most effective way to accelerate specific queries in a data warehouse. However, it only works when user queries can be correctly anticipated, which is not often possible [15]. Indexes also have limitations in what concerns their usage by the optimizer [9]. More recently, commercial systems have also added support for automatically picking indexes, provide tools to tune the selection of materialized views for a workload, and also tools that can recommend both indexes and materialized views [3]. It is worth noting that these techniques (indexes and materialized views) are general techniques that can (and should) be used with the data warehouse striping approach proposed in this paper.

There is a vast literature on query processing and load balancing in parallel database systems [e.g. 1, 14] and distributed databases [e.g. 17]. Many DBMS vendors claim to support parallel data warehousing to various degrees, e.g. Oracle8 [16], Red Brick Warehouse [20], IBM DB2 Universal Database [6], and the Advanced Decision Support Option of Informix Dynamic Server [13]. Similarly to our proposal Teradata system's parallel-processing architecture [7] divide a query workload among its processing nodes, which then execute the query in a true parallel and independent way. However, our approach is unique in what concerns the ability of using/sharing heterogeneous nodes and solving the unavailability of part of the nodes through the use of approximate answering techniques. Recently, there has been a significant amount of work on approximate query answering [2, 8, 11] where the main focus is to provide fast approximate answers to complex queries that can take minutes, or even hours to execute.

The technique presented in this paper - data warehouse striping - provides a flexible approach for distribution, inspired in both distributed data warehouse architecture and classical round-robin partitioning techniques. The data is partitioned in such a way that the load is uniformly distributed to all the available computers and, at the same time, the communication requirements between computers is kept to a minimum during the query computation phase.

This paper marries the concepts of distributed processing and approximate query answering to provide a fast and reliable relational data warehouse implemented with the existing computers of an organization.

3. The DWS-AQA System

The proposed technique uses the basic star schema approach to represent multidimensional data, distributing it over a set of computers that are available in the organization. This section discusses the major issues related to this topic. First, we describe the approaches used in DWS-AQA for setting up the environment and the minimal requirements that a computer must meet to be usable in DWS-AQA. Next, we discuss data loading, query distribution and result merging phases. These are very important aspects of DWS-AQA, as they support the distributed processing that lies in the kernel of the system, resulting in a very good speedup (typically DWS-AQA achieves a linear speedup of *N*, the number of computers in the system). The policy to manage the list of computers catalogued in DWS-AQA and to manage their inaccessibility or unavailability is also discussed. Furthermore, we show how DWS-AQA computes results to deliver fast early estimations of the final result using the partial results as they arrive from the computing nodes on the network. Additionally, to present the aggregation

results we propose an exploration interface that permits users to observe the progress of their aggregation queries. Statistical confidence intervals are also shown helping the user assess the proximity of estimated result to the final result. A periodic data load strategy is proposed, providing *24x7* availability which is an important requirement in some applications areas. Finally, we point out some of the limitations of the proposed technique.

3.1. Setting up the environment

The data warehouse administrator is responsible for creating the setup in all computers that compose the DWS-AQA environment. First, s/he must catalogue the computers that will be used in DWS-AQA, inserting their address in a hot list. Second, s/he is responsible to create the same star schema in all DWS-AQA computers.

The hot list contains relevant information about the DWS-AQA environment. Basically, each computer must fulfill two major conditions in order to be used in the DWS-AQA system:

- Memory, processor, and disk should have enough capacity to support a database engine. Modern computers can easily meet this condition.
- The normal computer utilization profile should not use all the resources (memory and processor) during all the time. It is worth noting that typical users normally use only a relatively small fraction of computer resources. Furthermore, most of the time the computers are simply not being used, either because the users are not connected or because the computer is idle waiting for user keystrokes.

One important aspect is that the use of the available processing capacity should not slow down the normal activity of the computer user. In principle, this can be achieved by a correct administration of the priorities given to the database background processes.

Another important aspect is that even when the computer user agrees to share the computer resources with DWS-AQA, there are still no guarantees that the computer will be available when a query is issued. There are many reasons for the momentary unavailability of a computer: it may be disconnected, have a failure, be temporarily too busy to provide a quick answer or network problems occur. The approximate query answering technique proposed in this paper solves these problems.

3.2. Data Loading Phase

In this section we show how DWS-AQA approach distributes the warehouse data over the catalogued computers. The data distribution used here is basically the same proposed originally in [5] and is summarized here to make this paper self-contained. The data is disseminated over the available computers according to the following guidelines:

- Dimension tables are replicated in each machine (i.e., each dimension has exactly the same data in all the computers).
- The fact data is distributed to the fact tables of each computer using a strict row-by-row round-robin partitioning approach (see also Figure 1). Each computer has *1/N* of the total amount of fact rows in the star schema, with *N* being the number of computers.

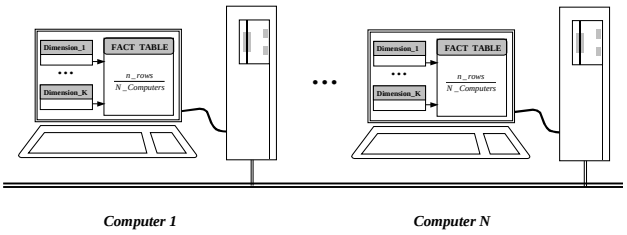

Figure 1. Data Warehouse Striping

This data partitioning for star schemas balances the workload by all computers supporting parallel query processing as well as load balancing for disks and processors. The replication of dimension tables doesn't represent a serious overhead because usually the dimensions only take less than 5% of the total required storage [18]. With this approach typical OLAP queries are executed in parallel by all the computers that constitute the DWS-AQA system.

An important aspect of the DWS technique is that it ensures a uniform distribution of fact data over all computers as this distribution is in fact a random sample method called systematic sampling [10]. The systematic sampling method has as sample basis one file or the elements list of the population, satisfying the hypothesis $M=N.n$, where M is the population size and N is a number that belongs to the natural number set (in our case represents the number of computers used by DWS-AQA system) and n is the sample size.

The procedure to apply the systematic sample method consists in choosing randomly one number k in the interval [1,N], which serves as seed and the first element of the sample. The systematic sample is composed by the elements with the following numbers k, $k+N$, $k+2N$, …, $k+(n-1)N$. If we have a population X_1, X_2, …, X_M and if we want to choose a systematic sample of this population ($M=N.n$), we can write

$$\begin{array}{lllll} X_1 & X_{1+N} & X_{1+2N} & \dots & X_{1+(n-1)N} \\ X_2 & X_{2+N} & X_{2+2N} & \dots & X_{2+(n-1)N} \\ \dots & \dots & \dots & \dots & \dots \\ X_N & X_{N+N} & X_{N+2N} & \dots & X_{N+(n-1)N} \end{array}$$

Each line is a systematic sample draw with *n* elements, selected from every N^{th} element. In our case, each of these lines represents the fact rows in each one of the *N* computers of the DWS-AQA system. This means the DWS technique uses a probabilistic sample method that is not biased due to its theoretical characteristics based on probabilistic theory. Another advantage is the possibility to compute the degree of uncertainty, i.e. the error of the estimate, in the form of confidence intervals. This is only due to the fact we have uniform random samples.

In [4] we have used real data from Dow-Jones stock index to show the round-robin data partitioning technique of DWS is uniform, i.e., the data is uniformly distributed over the computers. In the next section, we will see the query modifications required to distribute the queries over all the computers.

3.3. Query Distribution

DWS-AQA is a three-tier-architecture consisting of (1) Clients, (2) Query distribution and processing layer and (3) the various computers that represent the database nodes. The three-tier architecture is shown in Figure 2. Clients issue queries to the DWS-AQA system in SQL. The query distribution and processing layer is the heart of DWS-AQA system that provides a transparent interface to the data warehouse computing nodes. It is responsible for the following processes: query rewriting, query distribution and merging of partial results. The third layer consists of the set of computers that run the database independently of each other. In a DWS-AQA system the database nodes can also acts as clients, simultaneously.

Queries are distributed through all the computers that constitute the DWS-AQA system and their parallel execution by the available computers maintains the best load balance because the number of fact rows stored in each computer is about the same.

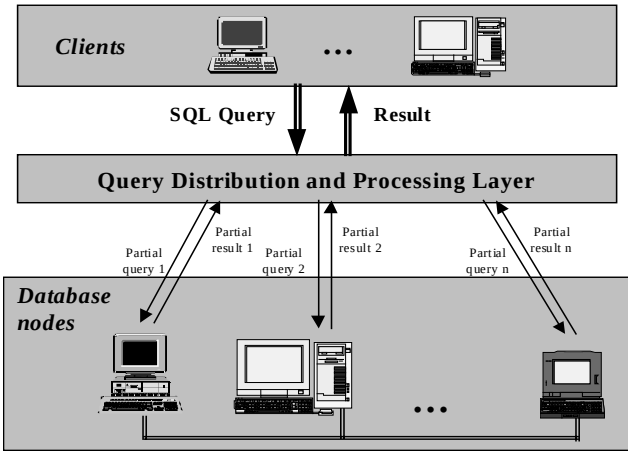

Figure 2. DWS-AQA Architecture

Most of the queries over a star schema can be transformed into *N* independent partial queries due to their nature (and because the fact rows are partitioned in a disjoint way: i.e., the rows stored in a computer are not replicated in the other computers). In a star schema, typical OLAP queries join and aggregate a large number of rows in the fact table and return few groups as final result. These queries use aggregation functions sum, count, average, standard deviation, and variance as operators. These are the functions that we study in our experiments.

3.4. Result merging of aggregation functions

In DWS-AQA system a query *q* is converted into a set of partial queries that compute partial results. These partial results are obtained independently in the different computers and sent to a coordinator computer where they are used for result merging. The result merging phase collects the partial results and computes running and final results for the user. The running results are estimations of the final result that are computed as soon as partial results arrive from each computer catalogued in DWS-AQA. These running results provide very accurate estimations even when some computers are temporarily busy or inaccessible. This section describes the merging strategy and the approach used to compute current estimations.

- **Computing SUM and COUNT functions**

If the original query contains the sum or count aggregation functions, then it is sent to all computers without modification (i.e. partial queries are exactly the same as the original query).

Now, we will see how DWS-AQA computes the approximate aggregation values when one or more computers cannot contribute to the final result. Consider the number of computers used in DWS-AQA to be $N = Nu + Na$, where Nu is the number of computers that are unavailable and Na is the number of computers that are available, contributing to compute the estimated aggregation value. The running results for the SUM and COUNT aggregation functions are basically computed as:

$$SUM_{estimated} = SUM_a + N_u \frac{SUM_a}{N_a} = SUM_a \frac{N}{N_a}$$

$$COUNT_{estimated} = COUNT_a \frac{N}{N_a}$$

where *N* is the number of computers used in the DWS-AQA system, SUM_a and $COUNT_a$ are the results of SUM and COUNT computed from the available computers. The attributes of the functions are not shown for simplicity. These formulas will be used in the experiments to compute the estimated value of final result.

- **Computing AVERAGE function**

Additionally, if the original query contains the aggregation function AVG, it must be modified before being sent. The modification are illustrated in this simple example:

```
select AVG(atr) → select SUM(atr), COUNT(atr)
from table        from table
```

If the aggregation function to compute is average and one or more computers are unavailable the running average is simply given by

$$AVG_{estimated} = \frac{SUM_a + N_u \frac{SUM_a}{N_a}}{COUNT_a + N_u \frac{COUNT_a}{N_a}} = \frac{SUM_a}{COUNT_a}$$

where SUM_a and $COUNT_a$ represents the partial sum and count from the available computers. Intuitively, the overall estimated average is the average taken from the available nodes.

- **Computing VARIANCE and STDDEV functions**

In order to compute the variance, the original query is transformed into a sum, count and average that is sent to each database node.

As before, the queries containing STDDEV and VARIANCE aggregation functions have to be rewriting following the rules illustrated in this example:

```
select stddev(atr)→select count(atr),var(atr),avg(atr)
from table          from table
select var(atr)→select count(atr),var(atr),avg(atr)
from table       from table
```

The running variance and standard deviation functions are simply evaluated from the partial results of the computers that respond or

$$VARIANCE_{estimated} = VAR_a$$
$$STDDEV_{estimated} = STDDEV_a$$

The previous formulas reveal us the modifications suffered by the original query before it is sent to all computers. While queries containing the aggregation functions SUM and COUNT, didn't require any modification the others functions (AVG, STDDEV and VARIANCE) need rewriting. The query distribution and processing layer is responsible for the transformation of the original queries and the merging of all the partial results, according to the operation, to obtain the final result.

- **GROUP BY and HAVING clauses**

The final result of queries containing GROUP BY and HAVING clauses is computed from the partial results by applying a query similar to the original to the partial results. The HAVING clause can be used to specify a further restriction over the groups. This clause only exists if we have the GROUP BY clause. In that case the HAVING clause is removed from the original query and is only applied after regrouping the partial results from all DWS-AQA nodes.

- **ORDER BY clause**

The final result of queries containing ORDER BY clause is computed from the partial results by applying a query similar to the original to the partial results. The ORDER BY clause, when it exists in the original query can also be sent to the DWS-AQA nodes. This means that each computer orders its partial set and the merging phase runs a faster algorithm to find the overall order from the partially ordered sets. This option means that each computer has more processing time but the merging phase has less work. In another way, sending the original query without ORDER BY clause meaning the result is only ordered in the merging phase. This has the disadvantage of more processing at final (merging phase). In both cases the results must be ordered when we merge the partial results.

3.5. Management of computer accessibility in DWS-AQA

DWS-AQA systems have a special computer, the controller, which has special requirements. The controller computer is used for specific tasks as to coordinate the operations in this environment but also acts as one of the nodes of the system. We describe how the controller manages a dynamic list of catalogued computers that can enter or leave the system as time evolves and how it manages the accessibility and availability of the DWS-AQA computers. The major tasks of the controller are described next.

- **Management of the list of computers catalogued in DWS-AQA**

The controller computer manages the list of computers catalogued in a DWS-AQA system. It is responsible to evaluate the accessibility of each computer during the query distribution and the loading phase and to manage these phases (it verifies if a computer is accessible or not). The inaccessibility circumstances are managed differently depending if it is a query distribution or loading phase. If a computer is not accessible during query distribution, the controller simply doesn't send the query to it. If this inaccessibility is verified during the loading phase, the controller is responsible to continue trying to load the data. The controller computer keeps track of the computers that are updated and where updates are not possible at this moment.

- **Management of the inaccessibility and unavailability of DWS-AQA computers**

The controller is responsible to check the accessibility of DWS-AQA computers before sending the queries and the hot list is updated accordingly. After that, the query is only sent to those computers that are accessible and the controller waits for the partial results. In this way, when partial results are presented, the estimations (running results) are computed. When new partial results are obtained these estimations are updated, providing increasingly more accurate results. At each point, the number of computers that have already contributed to the estimations is always pointed out (as will be described in the next section). For the "lazy" computers, we propose a timeout mechanism. This timeout can be set with a waiting time double to the last query response time. The user can always stop the processing or wait for the new partial results from the computers that didn't answer yet. If the user requests to stop the progress of execution, the controller sends a message to lazy computers to stop.

3.6. Exploration interface

In a DWS-AQA environment we expect the computers to answer the queries at different speeds. It is therefore necessary to offer an exploration interface, which shows current estimations continuously as the overall result is being computed. These estimations are more accurate each time a new computer contributes with new partial results. This exploration interface (see Figure 3) shows at least the query groups, the different estimations for each aggregation function and the respective confidence interval with a given probability, together with the indication of the fraction of computers which contributed and the corresponding degree of completeness of the query.

Such interface has some similarities with the one proposed in [12], which also provided continuous feedback to the user as the result is being computed. However, while their interface is continuously estimating the results as rows are fetched from one single source, DWS-AQA uses the interface to return early answers from the computers that have completed the processing while lazy ones are still working on the query. Furthermore, the user does not need to wait for the lazy computers, as the system typically returns very accurate estimations even when only a small set of computers have already answered (as we will see in section 4.3).

DWS-AQA online aggregation interface provides running aggregation results (**RESULT** column), i.e., an estimate of the final result based on the values returned by the computers that have finished their computation. The **Interval** column gives a probabilistic estimate of the current running aggregate to the final result.

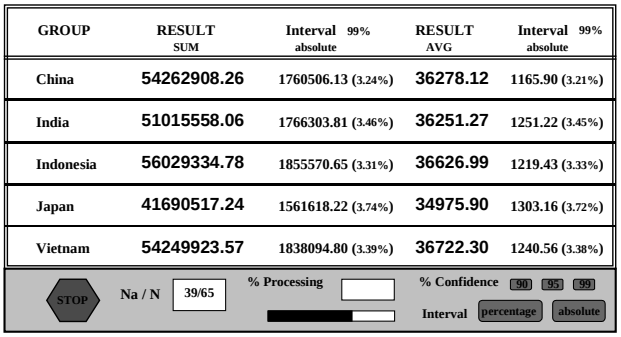

Figure 3. DWS-AQA Exploration Interface

At each time, the user knows how many computers answer (*Na*) and the total number of DWS-AQA computers (*N*). For example, according to Figure 3, the current average of India's revenue is within ±1251.22 of the exact result with 99% confidence. The interval 36251.27±1251.22 is called a running confidence interval. The size of such confidence intervals can be used as a measure of the precision of the estimator. We also give the possibility to display those intervals as a percentage, which approximately indicates the error that can be contained in the given approximate answer (these are the values shown within parenthesis in **Interval** columns). To compute confidence intervals we use formulas based in the standard central limit theorem that are based on the ones proposed in [4, 11].

3.7. Running the periodic load strategy that allows permanent availability

The controller computer runs a periodic data load strategy that allows permanent availability. Traditionally, data warehouses are refreshed periodically (for example, nightly) by extracting, transforming, cleaning and consolidating data from several data sources. During this period the data warehouse is unavailable for querying. However, there are business intelligence applications in telecommunications, electronic commerce, and other industries, that are characterized by very high data volumes and data flow rates, and that require continuous analysis and mining of the data. In the DWS-AQA environment we could provide a *24x7* availability. We propose a strategy where only one fraction of computers is updated each time and not all computers simultaneously. Two situations can be dealt with. First, if availability is not one of the main concerns in the organization, then all computers are loaded simultaneously (e.g. nightly loading without analysis). Second, if the organization requirements are *24x7* availability, then only 50% of the computers are updated each time, so that the system can still provide approximate answers during the update. This can also be done during the night where only the

computers that are being refreshed are offline but this requirement is not essential because DWS-AQA system continues working normally providing approximate query answering.

The controller computer keeps the information if a computer belongs to the fraction that is answering the queries or not (a query can be answered by either the computers already updated or the ones that are waiting to be updated). In those cases where the accuracy of the query result may be very important and approximate aggregates are not the solution, a mechanism that can be employed is the use of mirrored nodes.

3.8. Limitations of DWS-AQA technique and aspects not addressed in the present work

The proposed DWS-AQA approach seems to provide a very promising contribution to the scalability and efficient processing of huge data warehouses. However, the technique has some intrinsic limitations:

- DWS-AQA is specifically targeted to data warehouses organized as star schemas. This means that this technique cannot be arbitrarily adapted to other type of databases, in particular operational databases (OLTP). DWS-AQA is suitable essentially for those data warehouses that are predominantly dominated by one or more very large fact tables;
- Typically, the dimensions of a star schema are small in size when compared with the big fact table. However, there are exceptions to this rule, in which case the space overhead of DWS-AQA becomes more significant;
- Correlated queries and queries having sub-queries which use references from outer query blocks cannot be handled directly by this approach. To overcome this problem one solution is to use query de-correlation, as proposed in literature [19]. Using these techniques it is possible to rewrite the correlated query in such a way that outer references no longer exist. One problem with the rewriting strategy is that query de-correlation is not always possible and in some cases, although possible, it may not be efficient.
- The use of the available computers in a network raises very important security problems that are not addressed in this paper. Although this is not an intrinsic limitation of the DWS-AQA technique, as the use of security techniques to assure confidentiality is an orthogonal aspect of DWS-AQA (i.e., it does not interfere with the basic mechanisms of DWS-AQA), the actual use of DWS-AQA is clearly dependent on the assurance of the adequate level of security.

We are currently working further on these issues. Particularly, the security problem is one of the issues that is intensively being investigated, with particular emphasis on the study of the impact of the different security mechanisms on the DWS-AQA performance.

4. Assessing the speedup and accuracy of DWS-AQA

In this section, we present the results of an experimental evaluation of the DWS-AQA system. Using data from TPC-H benchmark [21], we show the effectiveness of DWS-AQA in providing speedup and highly accurate answers.

The rest of this section is organized as follows. We begin by describing our experimental testbed. We then evaluate the speedup of DWS-AQA and the accuracy of the approximate query answers as more computers contribute to the final result.

4.1. Experimental testbed

We ran the experiments on data from the TPC Benchmark™ H, with Oracle 8 as the back-end DBMS. TPC-H benchmark models a realistic business data warehouse, with sales data from the past seven years. It contains a large central fact table called **Lineitem** and several much smaller dimension tables. We used a scale factor of one for generating our test data. This results in a database size of approximately 1 GB. Table 1 summarizes some of the important features of the tables of TPC-H database used in the experiments.

Table 1. Characteristics of TPC-H tables

Table name	# of columns	# of rows
Customer	8	150,000
Lineitem	16	6,001,215
Nation	4	25
Orders	9	1,500,000
Region	3	5
Supplier	7	10,000

In the experiments we apply our technique to 100 computers, which corresponds to DWS-100. The use of *N* computers was simulated by dividing the *n_fact_rows* (6,001,215) of the fact table, using DWS-AQA technique, into *N* partial fact tables (LINEITEM_1,…, LINEITEM_N). Each computer has *n_fact_rows/N* rows and the dimensions are replicated in each computer. For example, DWS-100 simulates the use of 100 computers (*N*=100) having 100 partial fact tables (LINEITEM_1,…, LINEITEM_100) with each one having 60,012±1 fact rows, while the dimensions are equivalent to those of a

centralized data warehouse system. This simplification doesn't influence the results presented.

We are interested in typical queries that perform multiple aggregations and joins, processing a large number of rows of the fact table and returning a very small number of rows as result. We are also interested in analyzing the influence of group-by queries where the groups have different sizes. Therefore, in accordance with these criteria, we have chosen to implement query Q5 of TPC-H benchmark.

4.2. Speedup of DWS-AQA

Although this is not the main point of this paper, it is important to mention that DWS-AQA achieves an optimal speedup. We have made comprehensive experiments using 3, 5 and 10 computers with the same hardware characteristics [5]. Comparing the query execution time for a set of typical queries of a benchmark, we obtain an average speedup of 3, 5.1 and 11 using 3, 5 and 10 computers, respectively. In fact, the speedup is higher than the theoretical value, because the centralized data warehouse that was used as the reference experiment worked near the workstation memory and I/O limits. These results show that this technique can be applied to an arbitrary number of computers improving query execution speedup by almost *N*, the number of computers used. This is due to the fact that when we distribute the data we are working on more manageable data sets that could be more treatable in computers usually limited by memory space.

4.3. Accuracy of DWS-AQA

In this section, we consider a set of aggregate functions that are computed on the result of a complex select-join query. The query used is based on query Q5 of TPC-H benchmark that lists the revenue volume done through local suppliers. We modify it to return also the average, standard deviation and count of revenue. The query lists for each Nation in a Region the revenue volume (plus average, standard deviation and count) that resulted from Lineitem transactions in which the Customer ordering parts and the Supplier filling them were both within that Nation. The query considers only Parts ordered in a given year, displaying the Nations, and revenue volume, average, standard deviation and count ordered by nation name. The SQL statement for the query is:

```
select
     n_name,
     sum(l_extendedprice * (1 - l_discount))
as sum_revenue
     avg(l_extendedprice * (1 - l_discount))
as avg_revenue
     stddev(l_extendedprice * (1 -
l_discount)) as stddev_revenue
     count(*) as count
from
     customer, orders, lineitem, supplier,
     nation, region
where
     c_custkey = o_custkey
     and l_orderkey = o_orderkey
     and l_suppkey = s_suppkey
     and c_nationkey = s_nationkey
     and s_nationkey = n_nationkey
     and n_regionkey = r_regionkey
     and r_name = 'ASIA' and o_orderdate >=
'1994-01-01'
     and o_orderdate < ADD_MONTHS('1994-01-
01',12)
group by
     n_name
order by
     n_name;
```

This query is a 5-table join of large and small tables, where the data aggregated is reduced down to 1/5 of the Customers and Suppliers (representing one Region out of five) and 1/7 of the Lineitems (one year out of seven). The largest detail table has no direct selection applied to it. Five rows are returned constituting the revenue for each nation in the selected region:

```
N_NAME SUM_REVENUE  AVG_REVENUE  STDDEV_REVENUE  COUNT
-------------------------------------------------------
CHINA   53724494.26   36005.2483    21539.2708   1502
INDIA   52035512.00   36138.2886    22249.3627   1438
INDONESIA 55502041.17  36643.7337  22213.3269   1509
JAPAN   45410175.70   35227.3708    22113.7948   1288
VIETNAM 55295087.00   36755.0601    22519.8905   1506
```

In our experiments we evaluated the error obtained with this query for the SUM, AVG, STDDEV, and COUNT aggregation functions, when the number of computers that contribute to the final result is variable.

- **Error in estimated values**

When a query is sent to DWS-AQA system some of the computers could be too busy to answer or some problems in network arise. To simulate these situations we made experiments where the percentage of computers that are on-line varies from 10 to 90 percent and compute the relative error of the running results.

At each point the relative error of the running result is computed as:

$$error_{rel} = \frac{|exact\_value - estimated\_value|}{|exact\_value|} \times 100$$

The relative error obtained for Q5 query of TPC-H benchmark using DWS-100 and considering only the SUM aggregation function is shown in Figure 6 for each group (Nation). The *x*-axis represents the percentage of computers that are contributing to the final result.

In these experiments, the error decreases with the increment of the number of computers that are on-line (as expected). However, this error is not very large, as it does not exceed 15% with SUM aggregation function (not shown) and less than 20% with COUNT function.

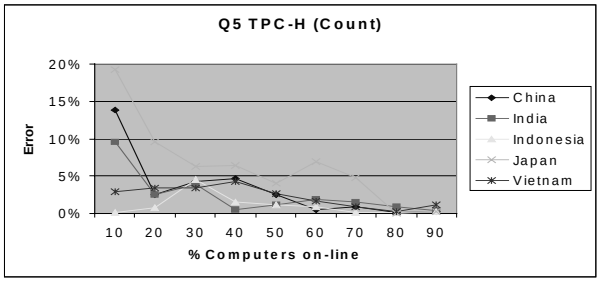

Figure 6. Revenue Error for SUM and COUNT

These are worst case results because only 10% of the computers have contributed with values. Furthermore, these errors are obtained for the smallest group, Japan, which aggregates only 104 elements. This also explains the oscillations in the error for this group.

Figure 7 shows the revenue error for Q5 query for all groups (Nations) when we are applying AVG function.

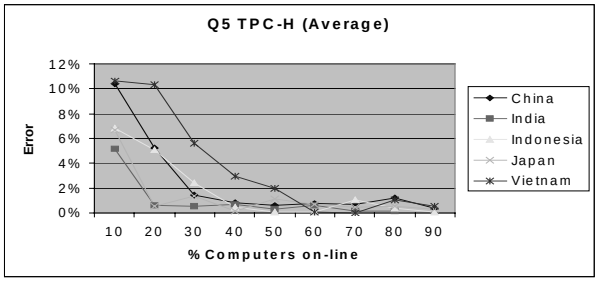

Figure 7. Revenue Error for AVG and STDDEV

In these cases the error obtained is even small, not exceeding 11%. And if we assume that half of the computers contribute to the result we can see from the figure that the error is always less than 2%.

The experimental results empirically demonstrate that DWS-AQA can give extremely quick answers with high precision for many queries and the user is also informed of errors incurred in the estimations by means of confidence intervals. Similar results were obtained for other TPC-H queries but due to space limitations we only show a synopsis of those results. Figure 8 shows the typical results obtained for queries Q1, Q6 and Q7 of TPC-H.

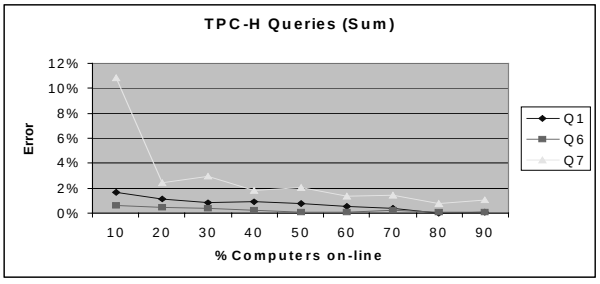

Figure 8. Error in queries Q1, Q6, Q7 using SUM

The queries Q1 and Q6 present better results due to the large number of tuples in the aggregated groups. The average number of tuples by group in queries Q1 and Q6 is 38,854 and 114,160 respectively, while in query Q7 the group only contains 1512 tuples. Since the groups are randomly distributed over 100 computers, it means that for query Q7, the individual computers only have 15 tuples in average, which results in less accuracy. Consequently, the accuracy of approximate query answers is highly dependent on the number of "samples" consisting of tuples from each group in individual computers.

- **Confidence intervals**

The confidence intervals give valuable feedback on how reliable an answer is. In the results we show four curves: the confidence interval limits (*lower_limit* and *upper_limit* curves), the estimated value of the aggregation function (*estimated_FUNCTION* curve) and also the exact value (*exact_FUNCTION*). The confidence interval indicates that the exact value (*exact_FUNCTION* curve) lies between the values represented by the *lower_limit* and *upper_limit* curves, with probability 95% or 99% (user choice). Figure 9 shows the 99% confidence interval for the China group of query Q5 using the sum aggregation function.

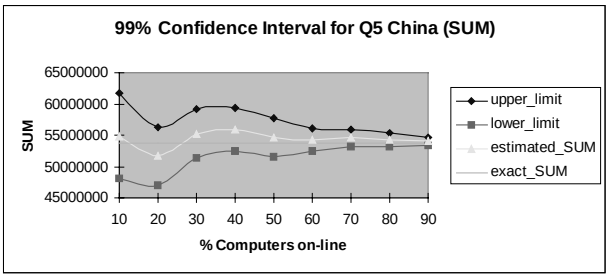

Figure 9. 99% Confidence Interval for query Q5, China group, using SUM

Figure 9 shows that the exact value is always within the limits of the confidence interval. The results also show that the magnitude of the confidence interval decreases, which means better precision, as the number of computers contributing to the running result increases. Figure 10 shows the results for the Japan group using variance aggregation function.

These experiments confirmed the effectiveness of using confidence bounds with the approximate results. The exact values (*exact_FUNCTION* curve) of the aggregation functions are always within the limits defined by the confidence interval and are very close to the estimated value (*estimated_FUNCTION* curve). Even when a large fraction of the computers didn't contribute to the results we obtain accurate approximate query answers.

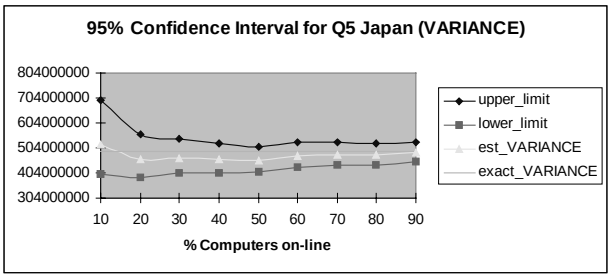

Figure 10. 95% Confidence Interval for query Q5, Japan group, using VARIANCE

In summary, using the techniques proposed in this paper, one can design a query answering system within the framework of the DWS-AQA system which not only produces accurate approximate query answers for complex join aggregates but also provides good error bounds using statistical techniques.

5. Conclusions

In this paper we proposed DWS-AQA as a technique to implement a large data warehouses using the available computers in the organizations. This approach, called data warehouse striping with approximate query answering (DWS-AQA), is based on the combination of two simple ideas: 1) uniform data striping to partition the data warehouse facts over an arbitrary number of computers, in such a way that queries can be executed in a true parallel fashion (a query is actually split into many partial queries), and 2) an approximate query answering strategy to deal with the momentary unavailability of one or more computers in the network. As the data warehouse shares computers that are also being used for other purposes, most of the times only a fraction of the computers will be able to execute the partial queries in time. However, as we show in the paper, the approximated answers estimated from partial results have a very small error for most of the plausible scenarios. Given the specific nature of the decision support activities, a small error is normally acceptable. Moreover, as the data warehouse facts are partitioned in a strict uniform way, it is possible to calculate tight confidence intervals for the approximated answers, providing the user with a measure of the accuracy of the query results. We also proposed an exploration interface to produce statistical confidence intervals for the estimated results, helping the user assessing the accuracy of the estimations. A periodic data load strategy that allows permanent data warehouse availability is also discussed. Experimental results show that the system returns fast and very accurate query results, while taking full advantage of available and inexpensive processing power in enterprise networked computers.

References

[1] Mahdi Abdelguerfi and Kam-Fai Wong. Parallel Database Techniques. IEEE Computer Society Press, 1998.

[2] S. Acharaya, P. Gibbons, V. Poosala, “Congressional Samples for Approximate Answering of Group-By Queries”, ACM SIGMOD Int. Conf on Management of Data, May 2000, pp.487-498.

[3] S. Agrawal, S. Chaudhuri, V. Narasayya, “Automated Selection of Materialized Views and Indexes for SQL Databases”, Proc. of the 26th International Conference on Very Large Databases, Cairo, Egypt, 2000, pp.496-505.

[4] J. Bernardino, P. Furtado, H. Madeira, “Approximate Query Answering Using Data Warehouse Striping”, Journal of Intelligent Information Systems, Kluwer Academic Publishers, accepted to publication.

[5] J. Bernardino and H. Madeira, “Experimental Evaluation of a New Distributed Partitioning Technique for Data Warehouses”, Proc. Int. Database Engineering & Applications Symposium, IDEAS’01, pp.312-321.

[6] S. Brobst, B. Vecchione, “Starburst Grows Bright, DB2 UDB”, Database Programming & Design, 1998.

[7] J. Catozzi, S. Rabinovici, “Operating Systems Extensions for the Teradata Parallel VLDB”, Proc. of the 27th Int. Conf. on Very Large Databases, Roma, Italy, 2001, pp.676-679.

[8] S. Chaudhuri, G. Das, V. Narasayya, “A Robust, Optimization-Based Approach for Approximate Answering of Aggregate Queries”, ACM SIGMOD Int. Conference on Management of Data, May 2001, pp.295-306.

[9] S. Chaudhuri and V. Narasayya, “An Efficient, Cost-Driven Index Selection Tool for Microsoft SQL Server”, Proc. of the 23rd VLDB Conference, Greece, 1997, pp.146-155.

[10] William G. Cochran. Sampling Techniques. Third edition, John Wiley & Sons, New York, 1977.

[11] P. J. Haas, “Large-sample and deterministic confidence intervals for online aggregation”, Proc. Int. Conf. on Scientific and Statistical Database Management, 1997, pp.51-62.

[12] 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.

[13] Informix Corp., “Informix Decision Support Indexing for the Enterprise Data Warehouses”, White Paper, 1998.

[14] H. Lu, B. Ooi, K. Tan. Query Processing in Parallel Relational Database Systems. IEEE Computer Society, 1994.

[15] P. E. O'Neil and D. Quass, “Improved Query Performance with Variant Indexes”, Proc. of ACM SIGMOD Int. Conference on Management of Data, 1997, pp.38-49.

[16] Oracle Corporation, “Star queries in Oracle 8”, White paper, 1997.

[17] M. Ozsu, P. Valduriez. Principles of Distributed Database Systems. Second edition, Prentice-Hall, New Jersey, 1999.

[18] T. Pederson, C. Jensen, “Multidimensional Database Technology”, IEEE Computer, December 2001, pp.40-46.

[19] J. Rao, K. A. Ross, "Reusing Invariants: A New Strategy for Correlated Queries", Proc. of ACM SIGMOD Int. Conference on Management of Data, 1998, pp.37 – 48.

[20] Red Brick Systems, Inc. “Star Schema Processing for Complex Queries”, White Paper, 1998.

[21] TPC Benchmark H, Transaction Processing Council, June 1999. Available at http://www.tpc.org/.