

# A middle layer for distributed data warehouses using the DWS-AQA technique

Marco Costa\*, Jorge Vieira\*, Jorge Bernardino+, Pedro Furtado\*\*, and Henrique Madeira\*\*

\* Critical Software SA, Coimbra, Portugal

+ ISEC-CISUC Polytechnic Institute of Coimbra, Portugal

\*\* DEI-CISUC University of Coimbra, Portugal

**Keywords:** Data warehousing, distributed query execution

## Abstract

The DWS (Data Warehouse Striping) technique is a round-robin data partitioning approach especially designed for distributed data warehouse environments. In DWS the fact tables are distributed by an arbitrary number of computers and the queries are executed in parallel by all the computers, guarantying a nearly optimal speed up and scale up. This technique is combined with an approximate query answering (AQA) strategy to deal with fails in one or more nodes. This paper presents a middle layer software and administration tools allowing the transparent implementation of DWS-AQA in commercial databases and OLAP (On-line Analytical Processing) tools. Specifically, the implementation for Oracle 9i and Discoverer is detailed, focusing the discussion mainly on the problems found in the process of turning a research result (DWS-AQA) into a commercial product.

## 1. Introduction

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.

DWS-AQA [1, 2] is based on the clever combination of two simple ideas: 1) uniform data striping (DWS) 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 (AQA) strategy to deal with the momentary unavailability of one or more computers in the cluster.

This paper presents a middle layer implementation of the DWS-AQA technique, currently under development at Critical Software SA, which allows the distribution of large data warehouses over an arbitrary number of computers (typically cheap workstations) and, at the same time, integrates the DWS-AQA approach with the data warehousing technology available in the market.

The paper is organized as follows. Section 2 briefly presents DWS-AQA. Section 3 presents the middle layer architecture. Section 4 describes the DWS-AQA administrative tools and section 5 summarizes the current state of the project.

## 2. The DWS-AQA technique

In the DWS-AQA technique the data of each star schema is distributed over an arbitrary number of workstations having the same star schema (which is the same schema of the equivalent centralized version). The dimension tables are replicated in each machine (i.e., each dimension has exactly the same rows in all the workstations) and the fact data is distributed over the fact tables of each workstation using a strict row-by-row round-robin partitioning approach (see Figure 1). 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 represent between 1 and 5% of the space occupied by all database [3]. The experimental results in [1] show that a DWS-AQA cluster can provide an almost linear speedup and scale up.

Typical OLAP queries are executed in parallel by all the nodes of the DWS-AQA system. If one node fails momentarily, the system computes an approximate answer and the confidence intervals based on the partial results obtained from the available nodes. Although the node failures represent exceptional events, even in a DWS-AQA cluster with many nodes, the approximate answer provided is normally useful for decision support, especially because the confidence intervals provide the user with a measure of the accuracy of the query results.

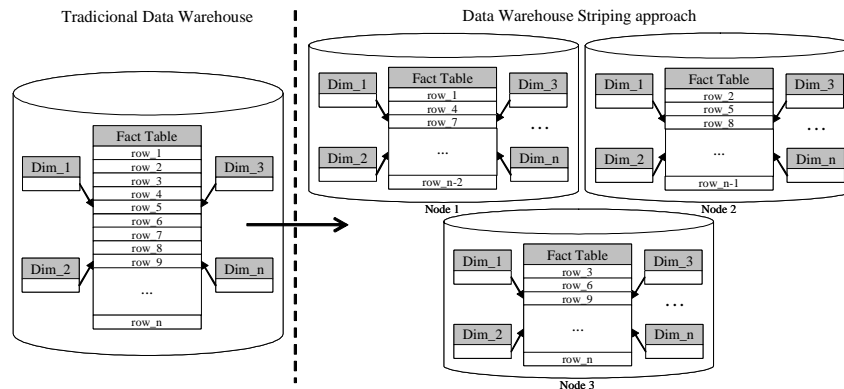


Figure 1 - DWS technique

### 3. Middle layer architecture

Before presenting the middle layer architecture let us briefly introduce the overall DWS-AQA system architecture (see Figure 2). A DWS-AQA cluster is composed by nodes running a DBMS and storing each one a fraction of the data warehouse partitioned according to the DWS approach. One or more nodes include the Controller SW, which is actually the DWS-AQA middle layer.

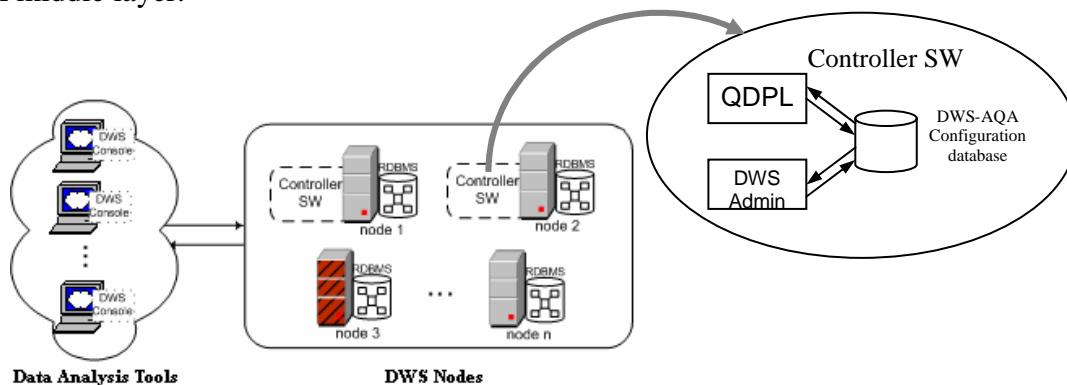
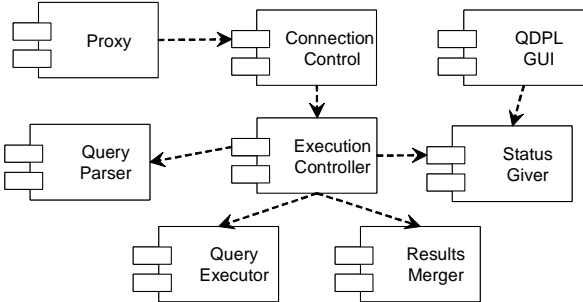


Figure 2 - DWS-AQA Architecture

The Controller SW comprises the QDPL (Query Distributed Processing Layer) the DWS Admin (Administration Tools), and a DWS Configuration Database. The DWS Configuration Database stores all the information necessary for the functioning of the DWS system and the DWS Admin is responsible for the administration of the DWS-AQA system.

The QDPL, which is the heart of the DWS-AQA technique, intercepts the OLAP queries sent by a commercial Data Analysis Tool, rewrites the queries if necessary, and controls the execution of the modified queries in the nodes. At the end, the QDPL merges the partial

results received from the nodes and returns the final result to the Data Analysis Tool. Figure 3 presents the QDPL architecture.



**Figure 3 - QDPL Architecture**

In short, the functions of each module are the following:

- **Proxy** - is responsible for receiving new connections to the DWS-AQA cluster, and for each connection it creates a new Connection Controller.
- **Connection Controller** - is responsible for controlling the connection between the Data Analysis Tool and the Data Base Server. It detects OLAP queries that are being sent over the connection, extracts them and creates a new Execution Controller component for each query received.
- **Execution Controller** - is responsible for executing the query in the DWS system. It coordinates the analysis and query rewrite performed by the Query Parser, the execution performed by the Query Executor and the construction of the final results performed by the Results Merger. It is also responsible for returning the available results, which are processed using the AQA technique, when needed. The Execution Controller also provides the status of the current execution to the Status Giver component and receives requests for aborting the execution from the Status Giver.
- **Query Parser** - is responsible for analyzing the query, and if necessary, rewriting it. For each query or sub-query the Query Parser constructs the three following queries:
  - Partial Query* – This query is executed in all DWS nodes creating partial results containing each node contribution to the final result.
  - Merge Query* – This query merges the partial results received from the nodes and constructs the final result for the current query or sub-query.
  - Merge Query AQA* – This query is very similar to the *Merge Query*, except that it is used to create the final result when one or more nodes are not able to provide an answer to the *Partial Query*.

Typically, an OLAP query is very complex has one or more sub-queries, creating a structure complex to analyze and transform into the Partial/Merge/MergeAQA queries described above. The Query Parser uses a recursive algorithm and a propagation mechanism to build an execution tree, where each element is a structure containing a Partial Query, a Merge Query, and a Merge Query AQA.
- **Query Executor** - is responsible for executing a given query in all DWS nodes.
- **Results Merger** - is responsible for merging the partial results into a table of a given node. If the results of one or more nodes are not available by the merging time, it should create a final result using the AQA mechanism.
- **Status Giver** - is responsible for broadcasting the execution status to the QDPL GUI and to the DWS Console.
- **QDPL GUI** - is responsible for the interface with the administrator of the system.

#### 4. Administration tools

The administration tools of the DWS-AQA system are divided in two subsets: configuration tools (Cluster Config, Star Schema Config, Parameters Config and Scripts Config) and maintenance tools (Load Control, System Availability Monitor, Statistics Analyzer, Reports Manager and Garbage Control). Very briefly, the role of each module is the following:

- **Configuration tools**

- *Cluster Config* - allows cluster configuration, including adding/removing nodes and changing the data load of each node (by redistributing the data among nodes).
- *Star Schema Config* - auto detects the star schema and identifies facts and dimensions.
- *Parameters Config* - allows setting various parameters needed for the correct functioning of the DWS System (e.g. use of AQA, maximum number of clients).
- *Scripts Config* - is responsible for managing all the database scripts used in DWS, including the generation of the scripts needed for the load of the fact and dimensions.

- **Maintenance tools**

- *Load Control* - is responsible for the data loads (initial and periodic) from the Data Staging Area into the DWS-AQA cluster nodes.
- *System Availability Monitor* - is responsible for periodically verify the availability of the cluster and log that information into the DWS-AQA configuration database.
- *Statistics Analyzer tool* - collects queries execution times in each node and proposes a new load balance to each node.
- *Reports Manager tool* - generates and manage availability and performance reports.
- *Garbage Collector* - works in background and is responsible for cleaning the various logs and temporary tables generated in the query execution.

#### 5. Current status and future work

The DWS project is in the implementation phase, and the first prototype of the commercial implementation of the DWS technique will be released in the third quarter of 2003. The first implementation of the DWS technique is for Oracle 9i database and Oracle Discoverer, however, the generic and portable architecture defined for this implementation will enable future migrations to other database servers and data analysis tools technologies.

The implementation of the first prototype revealed some interesting information regarding the types of queries that enable a best speed up. This fact indicates that for the future, a more powerful query rewriter should be considered. This implementation also revealed the need for powerful tools to adjust the load balance assigned to each node according to its historical performance logs and the characteristics of the nodes (e.g. CPU, memory, discs).

#### References

- [1] J. Bernardino, P. Furtado and H. Madeira, "Approximate Query Answering Using Data Warehouse Striping", Journal of Intelligent Information Systems, Vol. 19, N.º2, pp.145-167, 2002.
- [2] J. Bernardino, P. Furtado, H. Madeira, "DWS-AQA: A Cost Effective Approach for Very Large Data Warehouses", Int. Database Eng. & Applications Symposium, IDEAS'02, Edmonton, Canada, July 17-19, 2002.
- [3] T. Pederson and C. Jensen, "Multidimensional Database Technology", IEEE Computer, Vol.34, Nº12, December 2001, pp.40-46.