A MULTI-LAYERED ARCHITECTURE OF UNIFORMED DISTRIBUTED XOLAP CUBES (MAUDXC)

Developed By:

Saqib Subhan

307-FAS/MSCS/F06

Supervised by:

Dr. Sohail Asghar

Department of Computer Science

Faculty of Basic & Applied Sciences

International Islamic University Islamabad

2012

In The Name of

Allah Almighty

The Most Merciful, The Most Beneficent

INTERNATIONAL ISLAMIC UNIVERSITY ISLAMABAD
DEPARTMENT OF COMPUTER SCIENCE & SOFTWARE ENGINEERING

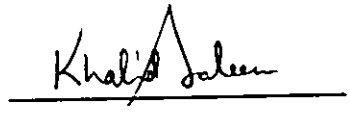
Dated: 24-01-2012

Final Approval

It is certified that we have examined the thesis titled "Multi-Layered Architecture of Uniformed Distributed XOLAP Cubes" submitted by Saqib Subhan, Registration No. 307-FAS/MSCS/F06, and found as per standard. In our judgment, this research project is sufficient to warrant it as acceptance by the International Islamic University, Islamabad for the award of MS Degree in Computer Science.

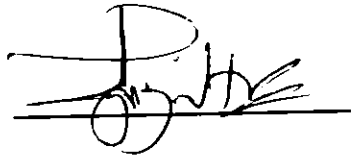
Committee

External Examiner



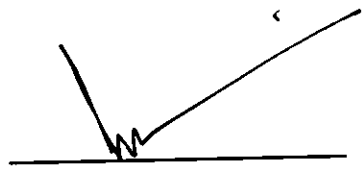
Dr. Khalid Saleem
Assistant Professor
Department of Computer Science
Quaid-e-Azam University, Islamabad

Internal Examiner



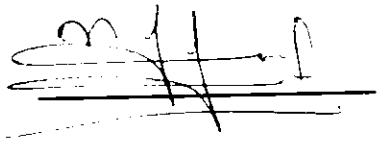
Dr. Ayyaz Hussain
Assistant Professor
Department of Computer Science & Software Engineering
International Islamic University, Islamabad

Supervisor



Dr. Sohail Asghar
Director / Associate Professor
UIIT, PMAS-Arid Agriculture University, Rawalpindi.

Co-Supervisor



Mr. Muhammad Imran Saeed
Assistant Professor,
Department of Computer Science & Software Engineering
International Islamic University, Islamabad

A dissertation submitted to the

Department of Computer Science,

Faculty of Basic and Applied Sciences,

International Islamic University, Islamabad, Pakistan,

as a partial fulfillment of the requirement for the award of the degree of

MS Computer Science

All My Words Dedicated With Respect and

Reverence Love and Affection to

"My Loving Family Specially My Parents"

&

"Teachers"

Their Love and Prays Always Accompanies Me

Like a Shining Star Whenever I was in Darkness

and Enable Me to Reach This Stage.

Dedicated To

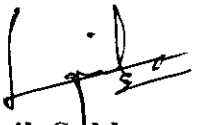
My Loving Wife

&

Sweet Daughter Umamah, Menaal & Loving Son Aayan

Declaration

I, hereby declare that "Thesis Topic" neither as a whole nor as a part thereof has been copied out from any source. I have developed this project and the accompanied report entirely on the basis of my personal efforts made under the sincere guidance of my supervisor. No portion of the work presented in this report has been submitted in support of any application for any other degree or qualification of this or any other university or institution of learning.



Saqib Subhan

Dated: _____ **307-FAS/MSCS/F06**

Acknowledgement

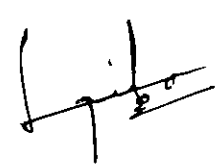
I have no words to express my deep sense of thankfulness to almighty Allah who bestowed upon me, the potential to complete this task.

First of all I would like to thank my supervisor **Dr. Sohail Asghar** who helped me out in every problem with his innovative approach, practical guidance and personal interest. Without his support and guidance it was not possible for me to complete my research work.

I would like to thank my internal supervisor **Mr. Imran Saeed** who was very helping and supportive and always provided right direction to me.

I express my heartiest regards to my Parents, Brother, Sisters and Wife. They supported me morally and prayed for my success to achieve goals in my life.

In last but not the least, I would like to thank to my friends specially **Mr. Asif**, **Mr. Saif** and **Mr. Sajid** for their help and cooperation.



Saqib Subhan

307-FAS/MSCS/F06

PROJECT IN BRIEF

Project Title	Multi-Layered Architecture of Uniformed Distributed XOLAP Cubes (MAUDXC)
Undertaken By	Saqib Subhan
Supervised By	Dr. Sohail Asghar
Starting Date	April 2009
Ending Date	May 2011
Software Used	C-sharp Framework 2.0, Oracle 11g Database Release 1, Analytical Workspace Manager (AWM) for Oracle 11g, Oracle Warehouse Builder (OWB) 11g, Oracle SQL Developer, Oracle Discover Administrator and Oracle Discoverer Desktop
Operating System	Microsoft Window XP
System Used	Pentium IV Core 2 Duo

ABSTRACT

Data sources in today's overwhelming business environment are often heterogeneous and sprinkled across several repositories. Data rise on scattered sources due to different geographical or business resources, furthermore, We use term multiple source data to encompass all cases where data are simultaneously obtain from multiple informants like data warehouses, Relational Database Management System (RDBMS), semi structural and unstructured format like XML and flat files.

Due to gigantic amount of data there is a complexity to get data in a standard format from several different sources. XML emerged as a standard language for describing and exchanging semi structured data from heterogeneous sources. With tags representing semantics of XML, the analysis of XML data is more flexible than relational databases therefore, OLAP on XML data (XOLAP) is desirable. An indispensable part for solving this problem is a standardized, vendor independent format "XCube" for describing multidimensional and OLAP data in XML format. The semantic heterogeneity problem can be resolved by creating X-Warehouses according to XCube standards. Data cube is a core operator for OLAP. Few techniques for XOLAP are available which generate data cubes from XML data or X-Warehouses. The limitation of XOLAP is that, it can generate cubes from either XML data source or X-Warehouses created on XCube standard. It doesn't support cube generation from heterogeneous data sources such as operational systems, data warehouses and flat files.

Therefore, extensions on XOLAP are desirable such as the ability to define data cube from heterogeneous data sources. Moreover, a uniform format is also required for these data cubes in order to avoid heterogeneity issues. The uniform format can be used for integration of these cubes.

This research work proposes the architecture for distributed XOLAP cubes with uniform format. The uniform representation resolves both technical and semantic heterogeneity problems. This architecture has the ability to acquire data from

heterogonous data sources for constructing ROLAP/MOLAP cubes. These ROLAP/MOLAP cubes are transformed into XOLAP cubes called X-Cubes for uniform representation. This architecture also supports the distributed cube by maintaining meta-cube at storage manager. In order to utilize these cubes for analysis, the architecture provides a query evaluator for efficient execution of OLAP queries on them.

The significance of this research is the presentation of the unique architecture for generating uniform distributed cubes of different interest from heterogeneous data sources along with the evaluation of OLAP queries against them.

TABLE OF CONTENTS

Chapter 1

Chapter 2

Chapter 3

Chapter 4

LIST OF FIGURES

LIST OF TABLES

Introduction

1. Introduction

This chapter discusses the concept of OLAP on multiple data sources along with brief introduction of data warehousing, OLAP, multilayered architectures, heterogeneous data sources, data cubes, ROLAP, MOLAP, XOLAP. This section especially focuses on computing OLAP on XML data (XOLAP), the integration and federation of XML and OLAP by highlighting some new research questions and their proposed solutions.

1.1. Data Warehousing and OLAP

Data warehousing and OLAP are very vital elements of a decision support system, which has become a focus of the database industry. All the major database system vendors are now providing the commercial products and services in this area. Decision support system imposes different requirements on database technology compared to traditional on-line transaction processing (OLTP) applications [14]. Data warehousing is a collection of decision support technologies, which enables the knowledge worker (executive/ manager/ analyst) to make better and faster decisions. The data warehousing technologies can be successfully deployed in many industries like retail, manufacturing, transportation, financial services, telecommunications, utilities, health care [14].

A data warehouse is a "subject-oriented, integrated, time-varying, non-volatile collection of data that is used primarily in organizational decision making" [36]. Due to multiple reasons, the data warehouse is maintained separately from the organization's operational databases. The data warehouse supports OLAP. The functionality and performance requirements of OLAP applications are quite different from OLTP applications which are based on operational databases [14]. Data warehouses are designed for analysis and decision support. It contains historical, summarized and consolidated data which is more important than detailed, individual records. Since data warehouses contain data from several operational databases, accumulated over long periods of time, they tend to be larger than operational databases; normally data warehouses are likely to be hundreds of gigabytes to terabytes in size. The workloads are mostly adhoc e.g. complex queries that can access millions of records and perform a lot of operations like joins, scans and aggregates.

Therefore, the query throughput and response times are more important factors than transaction throughput [14]. The multi-dimensional data is used in data warehouse for facilitating the complex analyses and visualization. Data warehouses might be implemented on ROLAP servers, assuming that data is stored in relational databases which support extensions to SQL. Moreover, it also supports the special access and implementation methods for efficient implementation of multidimensional data model and its operations. On the other hand multidimensional OLAP (MOLAP) servers directly store the multidimensional data in data structure like array and implement the OLAP operations over them [14].

1.2. Types Of OLAP

There are mainly two different types of OLAP: Multidimensional OLAP (MOLAP) and Relational OLAP (ROLAP). In MOLAP, data is stored in a multidimensional cube. The storage is not in the relational database, but in proprietary formats. MOLAP cubes are built for fast data retrieval, and are optimal for slicing and dicing operations. MOLAP are limited in handling of large amount of data because all calculations are performed when the cube is built, therefore it is not possible to include a large amount of data in the cube itself. ROLAP relies on manipulating the data stored in the relational database to give the appearance of traditional OLAP's slicing and dicing functionality. It has ability to handle large amounts of data but performance can be slow because each ROLAP report is essentially a SQL query (or multiple SQL queries) in the relational database, the query time can be long if it is applied on large databases.

1.3. Multi-Layered Architecture

A multilayered architecture is combination of different layers for allocating the responsibilities of an application. This concept applies to most real world organizations. Most of them work more or less the same way, they divide the tasks that are required in order to achieve the efficiency of the system.

1.4. Heterogeneous Data Sources

Heterogeneous data sources refer to the representation of data in different forms and formats. These formats can be in form of relational data, flat files, XML files and data warehouses etc. The integration of heterogeneous sources is always required for their simultaneous utilization.

1.5. Data Cubes

The data cube is convenient way of thinking about multiple aggregate views, all derived from a fact table using different sets of group-by attributes. A data cube allows data to be modeled in multiple dimensions for analyzing measures of interest. Usually measures are numeric values associated with business data. Data analysis normally includes dimensional reduction of the input data by using various aggregation functions [5]. Data cubes are popular in OLAP because they provide a natural way for analyst to navigate different levels of summary information in the database. The attributes in a data cube are categorized as dimension attributes, on which grouping may be performed, and measures are the result of aggregate functions [15].

Typical OLAP operations are rollup, drill-down, slice and dice and pivot. The rollup operation also called drill-up performs aggregation on data cube by increasing the level of aggregation. Drill-down operation is reverse of roll-up that decreases the level of aggregation or increases less detail data to more detailed data. The slice operation performs selection or projection on a single dimension of given data cube and produces a sub-cube while dice operation produces a sub-cube by performing selection or projection on more than one dimensions. The pivot also called rotate is a visualization operation used for re-orienting the multidimensional view of data [14].

1.6. OLAP on XML Data

Currently web is the largest data source so OLAP techniques on web data are required, because of its extensibility; XML (extensible Markup Language) formalism has emerged as a

dominant W3C standard in describing and exchanging data among heterogeneous data sources in a semi-structured way. Its simplicity and custom-defined tags make it usable as semantic preserving data exchange format. Its self-describing hierarchical structure enables a manipulative power to accommodate complex, disconnected, and heterogeneous data [12]

XML represents both structured and semi-structured data. XML is also an important standard of information exchange and representation on web. Information representation ability of XML is stronger than the traditional relational databases, but with tag representing semantics, XML data has more complex structure than relational databases. Analysis of data especially on web requires efficient data analysis techniques such as XOLAP (OLAP on XML data) to represent the complex semi-structure of XML data [7]

1.7. XOLAP

With increasing amounts of data being exchanged and even generated or stored in XML, a natural question is how to perform OLAP on XML data, which can be structurally heterogeneous (e.g., parse trees) and marked-up text documents [9]. Therefore a concept of XOLAP (OLAP on XML data) is introduced. In order to support XOLAP, a general aggregation operator of XML, GXaggregation is presented. Based on GXaggregation, XCube, an extension to traditional cube operator is defined on XML data model [7].

A core operator for OLAP is the data cube. While the relational cube can be extended in a straight forward way to XML, such extension would not address the specific issues posed by XML. In a relational warehouse, facts are flat records and dimensions may have hierarchies, but in an XML warehouse, both facts and dimensions may be hierarchical and XML is flexible because an element may have missing or repeated sub-elements. Moreover different instances of the same element type may have different structure. A cube Operator for XOLAP is presented called X^ 3 to meet the challenges introduced by these features of XML for cube definition and computation. [9].

Another data structure, IX-Cubes for XOLAP is presented to handles the difficulties due to semi structured nature of XML data [8]. A framework of XOLAP with broad set of OLAP operators (cube, rotate, switch, roll-up, drill-down, slice, dice, pull and push) with the support of TAX XML algebra is presented [11]. XCube, an open vendor independent and modular family of XML based document templates for describing and exchanging data cubes as they are common in warehouse systems. The most common structures are XCubeSchema, XCubeDimension and XCubeFact because these are most common ingredients of every multidimensional model [13].

An XOLAP is desirable for performing OLAP analyses over XML data. Although, few frameworks of XOLAP are presented in [7, 8, 9, 11], but none of them focused on uniform format for XOLAP, when the data sources are heterogeneous. An indispensable part for solving this problem is a standardized, vendor independent format (XCube) for describing multidimensional and OLAP data in XML format. The semantic heterogeneity problem can be resolved by creating X-Warehouses according to XCube standards [13]. Data cube is a core operator for OLAP. Few frameworks for XOLAP exist which generates cubes from XML data or XWarehouses. The limitation of XOLAP is that, it can generate cubes from either XML data source or X-Warehouses created on XCube standard. It doesn't support cube generation from other heterogeneous data sources such as data warehouses, operational systems and flat files. So extension on XOLAP is required which has the ability to define data cube from heterogeneous data sources. Moreover a unique format is also required for these data cubes to address the challenges arises due to heterogeneity of data sources.

We have extended the idea of XOLAP in this research work and proposed a novel architecture called Multi-layered Architecture of Uniformed Distributed XOLAP cubes (MAUDXC). The key to these XOLAP cubes is uniform representation. Our architecture is not limited to XML data sources only. This architecture generates uniformed cubes of different interest from heterogeneous data sources such as operational systems, data warehouses, XML and flat files. As XML is used as standardized format for data exchange among heterogeneous data sources, we are presenting a uniformed format of XOLAP cubes i.e. X-cubes. These cubes efficiently handle the heterogeneity issues. As the cubes are

distributed and belong to different domains, a query evaluator is proposed to handles these queries. This architecture also maintains the meta-cube which contains the description of different cubes stored locally or globally. The query evaluator first searches the meta-cube and then executes the query to appropriate cube for the result. The additional advantage of this architecture is that the idea of uniform cubes integration into a single global cube can be extended in two different ways. Firstly, global cube can be integrated from uniformed sparse domain cubes. Secondly global cube can be integrated from uniformed distributed cubes. These extensions can drastically enhance the capability of OLAP systems for answering complex OLAP queries.

This research work provides a significance contribution by presenting the architecture of distributed and standardized XOLAP cubes from heterogeneous data sources along with query evaluator for handling simple and complex OLAP queries.

2. Objectives

- To acquire standardized arrangement of data format in order to accomplish uniformity among manifold sources of data. XML can be used for this uniform format.
- There is no such architecture which provides consistency due to uniform data cubes for handling OLAP queries from heterogeneous data sources. Therefore, need arises for an architecture, where uniform cubes of different domains are generated to fulfill the analysis requirements. For achieving this, the architecture also requires query evaluator for executing the OLAP queries on these cubes.
- The idea of uniform cubes can be used to integrate the global cube for answering the complex OLAP queries.

3. Organization Of Study

The rest of the study is organized as described under:

- **Chapter 1** provides the brief overview of data warehousing, OLAP, OLAP on XML data, OLAP and XML integration, OLAP and XML federation.
- **Chapter 2** reviews the previous research work that contributes to the integration of XML and OLAP by computing OLAP data cubes and identifies the limitations in previous study.
- **Chapter 3** outlines the problem statement and the requirements of this research work.
- **Chapter 4** proposes a Multi-Layered Architecture of Uniformed Distributed XOLAP Cubes (MAUDXC) that can effectively deal with the problems in existing systems. This chapter also describes each component of proposed architecture along with their functionality and methodology.
- **Chapter 5** describes the scope of the research work
- **Chapter 6** provides the validation and evaluation of the proposed architecture MAUDXC through different experiments and results performed on real and test datasets
- **Chapter 7** concludes the efforts performed in this research thesis along with providing future directions for further research.

4. Terminologies

The list the different terminologies / acronyms used in this thesis are as following:

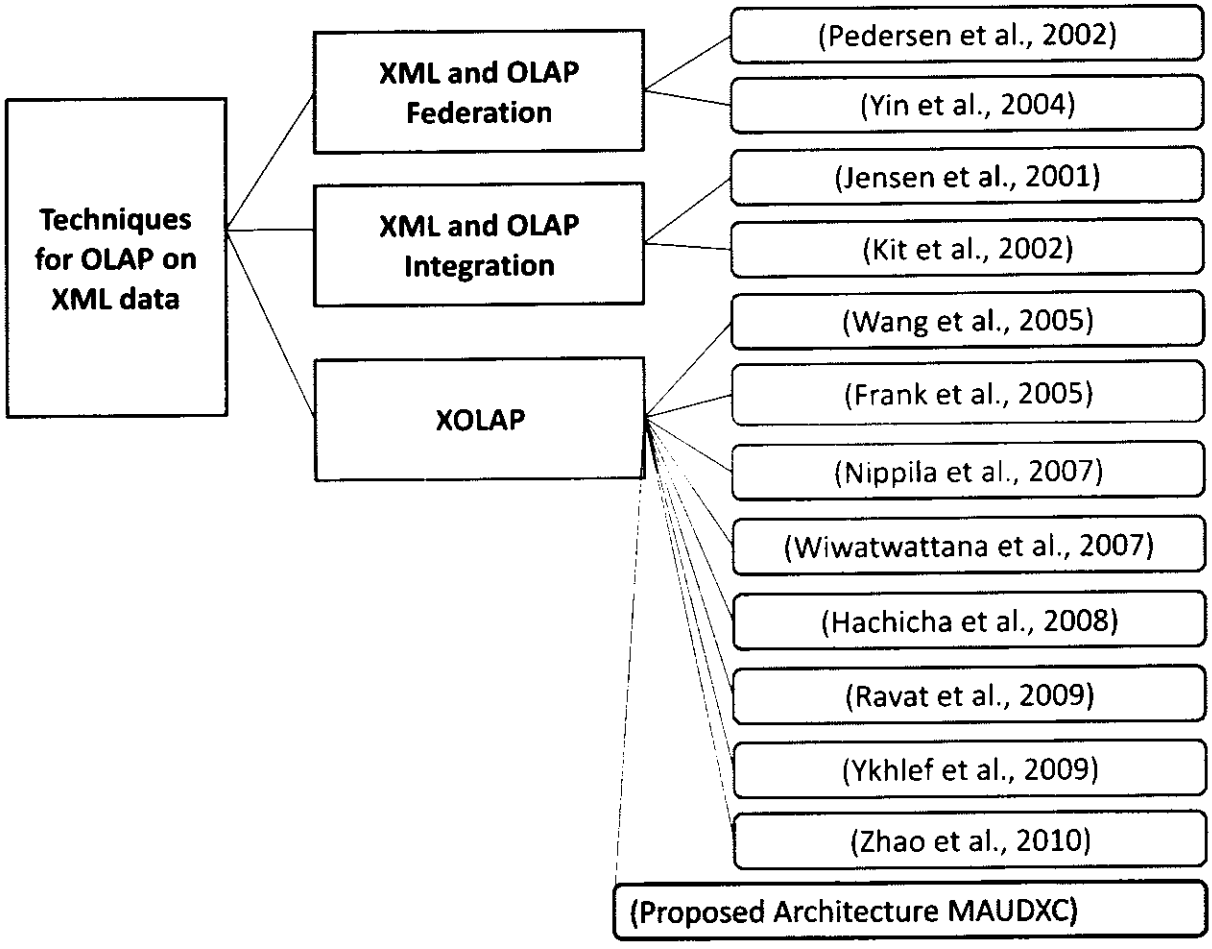
OLAP	Online Analytical Processing
OLTP	Online Transactional Processing
XML	Extensible Markup Language
XOLAP	OLAP for XML data
MAUDXC	Multi-Layered Architecture of Uniformed Distributed XOLAP Cubes
MDX	Multidimensional Expressional Language
AWM	Analytical Workspace Manager
OWB	Oracle Warehouse Builder
GDD	Global Data Dictionary

QA	Query Analyzer
QM	Query Matcher
QE	Query Evaluator
QMx	Query Modifier
QEx	Query Executer
OQ	OLAP Query
MQ	Modified OLAP Query
UIMS	User Interface Management System
ERRA	Earthquake Reconstruction and Rehabilitation Authority
CSV	Comma Separated Values

Literature Review

2. Literature Review

The objective of this study to review the previous work that contributes to the different techniques used for analyzing OLAP on XML data. Many of the researchers have worked on these techniques. The limitations in their proposed concept in different aspects are discussed in this section.



2.1 Literature Review

Pederson et al. [2] has suggested a model for federating OLAP data with external XML data. He also recommended the need for efficient query processing for this federation. They presented three effective cost based optimization techniques namely as in-lining, XML data retrieval though limited interface, caching and pre-fetching. In amassing to that they have

also utilized some experimental approaches that show the techniques to get the optimal solutions. Lastly, caching and pre-fetching have been performed on XML and OLAP cubes and results have been stored in fact table [2]. The idea of XML and OLAP federation supports the logical integration between OLAP and XML data. Hence, overcome the limitations of physical integration. The introduction of these novel techniques is a major research contribution because cashing and pre-fetching drastically improves querying execution time thus improving the query performance greatly. The limitation of their work is that Top-down and bottom-up in-lining strategies are used for optimization, but both fails to find the optimal solutions in certain situations like both techniques generate in-lining expression for all combinations but high cost combinations are not considered. Bottom-up approach has fewer chances to fail because it first considers the strategy where all expressions are in-lined.

Pederson et al. [3] has worked for improving the robustness of their federated system by presenting some important techniques like OLAP components, relational components and auxiliary components. These techniques improve the reliability of external XML components by allowing multiple valid data sources to be used, possibly including outdated backup sources that may be utilized if none of the sources are accessible. By using multiple sources and then selecting the fastest will enhance the performance of federated system in accumulation to the improved reliability. This provided two major research contributions. Firstly to provide an overview of integrated solution to the problems that may arise in OLAP-XML federations, including data changes in OLAP sources. Lastly present a technique for improving the reliability of external XML sources. Major strength of their work is that exploitation of logical data integration will increase performance. The employment of basic components reduces average retrieval time of data from distributed sources. The limitation of their work is that the approach of allowing several data sources may not be sufficient and fails if the XML document is removed permanently and no alternate exists. The handling of more than one data sources may cause problems with consistency of data.

Yin et al. [4] extended the previous work on OLAP-XML federations in different ways. Firstly, proposing simplified logical query semantics with decoration operator, federation selection operator and federation generalization projection operator which yields more compact and concise logical query plans for SQL*XM* queries. Secondly, a set of physical algebra with cube selection, cube generalization projection, fact-transfer, dimension-transfer and XML-transfer operators is offered in order to model the actual query execution tasks precisely. Thirdly, query plan optimization is performed in four phases: query rewriting, logical query conversion, cost estimation and plan space pruning. Fourthly, the implementation of a robust query engine is described, which generates component queries for underlying data sources and evaluate the final execution plan in a bottom-up manner. Finally, a performance study has been performed that shows the effectiveness of the approach [4]. The contribution of their work is to present the extended version of logical integration between OLAP and XML Data by providing XML-extended OLAP queries based on physical algebra. Few limitations of this approach are as following:

- They need to develop more advance query optimization technique in order to improve the query engine.
- They are required to apply a more efficient OLAP and XML data loading technique, query evaluation technique and that of cost estimation technique in order to improve the performance of the system.
- Also, more performance studies should be performed to reveal the behavior of the OLAP-XML query engine.

Kit et al. [23] have been proposed an XML-OLAP system based on relational databases for the analysis of XML data. In addition to that they have developed some algorithms for topological rollup which is structured based grouping in XML-OLAP. These algorithms help to speed up the computation and bases on structural join algorithms. They also proposed more efficient Single-Scan by preorder/ postorder algorithm which enables to compute the structure-based grouping through a single scan over stream of XML nodes. Finally they have applied these algorithms on very large XML data aggregation for achieving information. Main strength of their work is that, due to utilization of topological algorithm will improve the efficiency of the system. Limitation of their work is that topological operator's

enhancement is required for logical integration of the proposed system along with applying partitioning technique to the proposed algorithms.

Jensen et al. [27] presented an approach for specifying OLAP cubes on XML data without their physical integration. They proposed a multidimensional model called UML snowflake diagram that provides the precise specifications of OLAP database containing multiple XML or Relational data sources. UML used the document type definition (DTD's) for describing and visualizing the logical structure of XML documents which eases the design of OLAP DB. They presented the architecture that integrates the XML data at conceptual level. Their proposed model also supports the special considerations (handling dimensions with hierarchies) that need to be taken when designing an OLAP DB on top of XML data. The strength of their work is that the presentation of a new multidimensional model for integrating XML and Relational data sources at conceptual level by constructing UML snowflake diagram, which eases the design of OLAP DB's along with the support of dealing the special issues of dimensions with hierarchies in OLAP. The limitation of their work is that the approaches presented in this paper need to be implemented along with the efficient query processing techniques such as query translation.

Bordawekar et al. [5] proposed a logical model for XML analysis based on the abstract tree-structured XML representation. The traditional multi-dimensional OLAP model is unsuitable for XML analysis due to its weakness in representing semantically-rich, semi-structured XML data. Moreover this logical model demonstrates the practicality of the new model by proposing new structural aggregation extensions for XQuery. New aggregation operators are structural group-By, topological roll-ups and topological cube [5]. The introduction of logical model for XML analysis is their major research contribution. The main strength of their work is that the proposed abstract XML tree-structured model can represents semi-structured data and support analytical operation on them. This model also supports order oriented and value structural queries over numeric and non-numeric measures. Major challenges of their work are that they are required the implementation of aggregation operators in the XQuery framework, complexity in creating views for large number XML documents and visualizing large hierarchical data sets.

Park et al. [6] proposed a model of XML-OLAP for multidimensional analysis of XML warehouse. They represented both dimension and fact data are as XML documents [6]. They relied on an index structure for matching each dimensional data with corresponding index data. They utilized conceptual model by using UML classes in order to build XML documents [6]. Consequently, they developed a new language by integrating XML and MDX called XML-MDX. Later on, they also developed XML cube named as XQ-cube, which is efficient in representing XML-OLAP warehouse [6]. The strength of their work is that their proposed model enables easier collection and analysis of its documents. Moreover, XQ-cube is a generalization of relational cube that is used for both numeric as well as textual data representation. Limitation of their work is that when documents are merging to develop an OLAP cube the problem of document ordering makes difficult to define optimize query for OLAP.

Wang et al. [7] presents the concepts of XOLAP, OLAP of XML data. A general aggregation operator GXaggregation is presented for supporting XOLAP. On the basis of GXaggregation operator, another Operator XCube is defined which is an extension of tradition cube operation. Both operators can be embedded into XML query language such as XQuery. XOLAP can be used for analysis of XML information in XML warehouse or information integration system. The operators of XOLAP can easily be embedded in existing XML query languages. Based on the traversal of XML tree, these operators in XOLAP are implemented by GXaggregation algorithm [7]. Stack-list is the main data structure used by this algorithm for representing aggregation nodes in traversed path containing three parts: aggregation result, dimension list and level table. The major contribution of their work is the introduction of recursion concept for recursive aggregation. Traditional OLAP is defined with flat structures on relational or multidimensional data. Dimensions may have hierarchy even with star or snowflake schema and the instances with recursive measures are not handled. While in XOLAP, XML has hierarchy structure. Recursion is permitted that allows recursion aggregation. The limitation of their work is the usage of many storage structures; choice for selecting the storage structure adaptive for XOLAP is the problem which is not handled.

Jian et al. [8] proposed a new data structure of IX-cube (Iceberg Cube) over XML data. The IX-Cube and techniques used for computing data cubes from XML data having semi-structure nature, is quite different than computing data cubes on relational data. They also proposed two algorithms RCube and BUC for computing the IX-Cubes [8]. The OLAP queries can be answered by performing roll-up and drill-down operations on IX-cube. Major contribution of their work is the conversion of data cube from relational data to XML data and then enabling proposed data structure for efficient handling of OLAP queries. Both cell queries and sub-cube queries are used and B+-tree CubeIndex applied for acceleration of query answering [8]. Main strength of their work is that by utilizing the tree structure in IX-cube for representation of XML data may make easy increment and updating of dimensions and measures in the cube.

Wiwatwattana et al. [9] presented a concept for dealing with the computational and semantic challenge arises for an XML data cubes. They proposed a new definition for cube operator (X^ 3) for XML data which includes a suitable generalized mechanism for specification by providing a new algorithm. The XML tree pattern relaxation techniques have been utilized in order to handle these challenges. They developed a relational cubing algorithm in order to implement a summerizabality property for XML data. They also developed group of cube computations algorithms such as counter base, bottom up and top down algorithms [9]. Thus, summarizability properties implement these algorithms to get efficiency in XML and OLAP cube data [9]. These algorithms allow two kinds of summarizability violations like nondisjointness of grouped partition and incomplete coverage of dimension attribute [9]. Main contribution of their work is the introduction of X^ 3 cube lattice properties and cube computation algorithms for implementing the summarizability property to get efficiency. Limitation of their work is that they are unable to automated system with the purpose of determining cube lattice properties.

Hachica et al. [11] provided a XOLAP (OLAP over XML data) framework. They used Tree Algebra for XML (TAX) in order to implement XOLAP. They also focused on construction of OALP analyzer over XML cubes named as XOLAP. Furthermore, they

presented XOLAP operators in TAX tree in three different categories. The categories of these operators are structural operators (rotate, switch, push and pull), set operators (slice and dice) and granularity-related operators (roll-up, drill-down and cube) [11]. They also compared TAX, XOLAP and X^3 to achieve a full XOLAP environment [11]. The major contribution of XOLAP framework is that, it supports integration of TAX operators with XOLAP to achieve better performance and efficiency. The limitation of their work is that the efficiency of XQuery is limited when user wanted to compute complex analytical queries. Moreover, XOLAP and TAX expressions are basically unable to generate automatic optimal XOLAP queries.

Hummer et al. [13] proposed a unique, standardized and vendor independent format for describing the multidimensional data so that, the exchange of data between heterogeneous systems can be done in much easier way. They presented XCube [13], an open, manufacturer independent and XML based document template for storing, exchanging and querying the OLAP data. The most important structures of XCube are XCubeSchema, XCubeDimension and XCubeFact [13]. These are also the common ingredients of every multidimensional model. These structures are flexible enough to support special features like shared roll-ups or multi-cubes. By splitting the description of a data cube into three parts, a client can easily determine the certain cube of his interest, due to the smaller size of schema and dimension data than the huge amount of fact data. The contribution of their work is that XCube completely solves the technical heterogeneity problem when integrating the several Data Warehouses. The semantic heterogeneity problem can also be solved by creating a new data warehouse according to XCube standards. This provides an easier way to integrate this data into other warehouses. Standard dimension can be introduced similar to standard time dimension. Limitation of their work is that XML documents tend to be rather large by using XML for exchanging data cubes.

Frank et al. [19] proposed an approach for integrating semantically related data from heterogeneous data sources using XML techniques. This approach basically transformed all data cube into XML formats [19] in order to amalgamate data into global virtual cube for sharing the data over the internet by using XQuery. Main strengths of their work are

transformation of all cube in to XML formats. will resolves the semantic discrepancies and the integration of data into global cube through XQuery utilizes the integration power of internet. Main limitation of their work is that data cube size is very large which will make the integration process time consuming. The proposed technique doesn't cover the scope of data cubes from different domains. The support of query evaluator is also lacking in the proposed architecture which can enable to answer the complex OLAP queries.

Nappila et al. [21] have been recognized capabilities of XML, which greatly supports data sharing by providing a standards data exchange format for heterogeneous data sources. Furthermore, they applied an XML language for construction of data cube in structurally heterogeneity of XML environment. Basic strategies introduced are LCA (Lowest Common Ancestor) and SPC (Smallest Possible Context) for query evaluation. In accumulation with this they have demonstrated a prototype for OLAP cube construction and data integration. XML query languages XQuery and Xpath have used for query and integration of heterogeneous data. Strengths of their work is that the proposed system gathered data from heterogeneous sources and tight them under a single format of XML which makes system efficient. Limitation of their work is that the heterogeneity problem particularly semantic and syntactic is not completely handled. Moreover XQuery does not provide high level primitives for construction of OLAP cube.

Pedersen et al. [22] presented an approach for the federation of OLAP and XML data. given in terms of a formal data model and algebraic query language. To demonstrate this approach a federated query language is introduced called SQL-XM, incorporating the XML query language X-Path into a subset of SQL adapted to multidimensional querying. SQL-XM allow XML data to be used directly in an OLAP query to decorate multidimensional cubes with external XML data, and to group and select cube data based on XML data values. The incorporation of XML data in cubes was made such that semantic problems were avoided, e.g., when aggregation was performed on the resulting cube no double-counting of data could occur. A number of effective optimization techniques for OLAP-XML federations were described. Finally, a prototype implementation and a set of experiments stating the

performance of the approach and the effectiveness of the optimization techniques were presented, showing the attractiveness of the approach compared to physical integration. Major contribution of their work is the concept of the logical federation of OLAP and XML data sources instead of physical integration that is time-consuming process. This logical federation also handles the special issues like dimension hierarchies and correct aggregation of data. Limitation of proposed model is that the transport of data from OLAP and XML components may becomes the primary bottleneck in the federation. Therefore, efficient optimization techniques are required to overcome this problem. Another limitation is the proposed query language SQL-XM will perform satisfactory for only small databases.

Niemi et al. [25] presented a method for facilitating the analysis of large distributed heterogeneous data sources. This method helps the user to construct an OLAP cube from distributed data according to the analysis requirements. This method utilizes the XML language and provides an XML presentation for OLAP cubes and its schema. Hence, an OLAP cube in XML form can be easily transformed into a form that OLAP servers can read. Finally, they analyze the actual data by using an OLAP server product. The strength of their work is that distribution of computation can speed up the building process of OLAP cube and increasing the system's performance. Contribution of their work is that they achieved correct aggregations in the presence of dimension hierarchies but some of these results can be applied to distributed aggregation calculation, too. The limitation of their work is that dimension hierarchies are partly normalized that may contain lots of redundancy.

Ravat et al. [30] proposed a multidimensional OLAP model called Galaxy model for the analysis of XML documents. This model enables the analysis of XML document and their metadata with the use of numeric and textual indicators [30] and doesn't restrict the decision makers for limited and predefined analysis. The dimensions with multiple hierarchies allow the representation of document structure, contents and metadata along with their links. So this model is capable to analyze data sources and requirements of decision makers for providing valid OLAP schemas. In accumulation to this model they also implemented a CASE tool for the design and implementation of galaxy schemas. The tool

utilizes the graphical interfaces for providing the representation of multidimensional schema and its associated loading processes. The contribution of their work is as followings:

- Proposed a model that supports the availability of multidimensional elements representation, for expressing analyses.
- Presented an approach that integrates the XML documents within the decisional system.
- Provided a tool to ease the design and implementation process.

The limitation of their work is that, they have not utilized the XML documents with both DTD and XML schemas. They are also required to implement the query optimization techniques for speeding up the processing of XML data.

Ykhlef [32] proposed a XML data cube model which is a well-founded approach for representing OLAP data by using XML. These XML data cubes allow the selection and aggregation operations on external XML data. They also described that how OLAP queries on XML data cube can be expressed in a natural way by using SQL-like query language called XRQL. This will enables the analyst to gain insight into data through fats access to a variety of data views on data. Moreover they presented the extensions in XRQL language by proposing GROUP BY ROLLUP and GROUP BY CUBE operators. The utilization of these operators along with query nesting technique will enable the analyst to express more complex OLAP queries over XML data cubes. The contribution of this work is three fold. Firstly, an XML cube model is presented. Secondly, how OLAP queries on XML data can be expressed in XRQL. Finally, an extension in XRQL is presented for expressing more complex OLAP queries.

Zhao et al. [33] presented a XML cube model called XMLCube for OLAP on XML data. This model is based on indexed structure used for aggregation query from XMLCube. They also proposed an index method of the cube for group by aggregation query. The index structure utilizes the hash and tree indexes to quickly accomplish the measurable path in the cube. The algorithms for building the new index and corresponding group aggregation query in an XML data cube is also presented. Finally, they implemented these algorithms for

estimating the cost of the query. The major contribution of their work is that they focused on solving the problem of low efficiency of aggregation queries over XML cubes. The limitations of their work are they didn't address few technical problems like controlling redundancy in the index and size of the index.

2.2 Critical Evaluation

This segment first provides the critical evaluation of the previous work discussed in literature review. This section also compares and contrasts the above given literature review and our proposed architecture. Table 2.1(a) and 2.1(b) will provide the criteria based comparison between integration techniques of XML and OAP.

Pederson et al. [2], [22] presented the federation of OLAP and XML data, given in terms of a formal data model and algebraic query language. For demonstrating this approach a federated query language is introduced called SQL-XM, incorporating the XML query language X-Path into a subset of SQL adapted to multidimensional querying? SQL-XM allows XML data to be used directly in an OLAP query to *decorate* multidimensional cubes. The idea of XML and OLAP federation supports the logical integration between OLAP and XML data. Hence, overcome the limitations of physical integration. They presented optimization techniques for efficient query processing for XML and OLAP federation. These effective cost based optimization techniques namely as in-lining, XML data retrieval though limited interface, caching and pre-fetching. Yin et al. [4] also applied the similar concept by extending the OLAP-XML federations in different ways by providing XML-extended OLAP queries based on physical algebra. They also applied algebra based query optimization and evaluation techniques and the foundation of a robust federation query engine with query plan generation. Park et al. [6] also presented the similar idea by proposing a model of XML-OLAP for multidimensional analysis of XML warehouse. They developed a new language by integrating XML and MDX called XML-MDX. Later on, they also developed XML cube named as XQ-cube, which is efficient in representing XML-OLAP warehouses. The similar concept is implemented by Wang et al. [7]. He worked on XOLAP, OLAP of XML data by presenting a general aggregation operator for supporting XOLAP. Hachica et al. [11] also

applied the same idea by providing a XOLAP (OLAP over XML data) framework. They used Tree Algebra for XML (TAX) in order to implement XOLAP. They also developed an OALP analyzer over XML cubes named as XOLAP.

By working on same idea of integration Kit et al. [23] proposed an XML-OLAP system based on relational databases for the analysis of XML data. In addition to that they have developed some algorithms for topological rollup which is structured based grouping in XML-OLAP. Vrdolijak et al. [17] applied the concept of logical integration of XML data through XML schema representation in order to build a web warehouse and Frank et al. [19] also implemented the approach of semantically related data integration from heterogeneous data sources using XML techniques. Jensen et al. [27] presented an approach for specifying OLAP cubes on XML data without their physical integration. They proposed a multidimensional model called UML snowflake diagram that provides the precise specifications of OLAP database containing multiple XML or Relational data sources.

By making a comparison, the similar work is done by Yin [4] and Pederson [2], [22] by providing novel techniques of the query optimization of XML-OLAP federation. Pederson [2], [22] implemented these techniques which greatly increases the query execution time and performance along with the efficiency of the federation but both implemented methods are not providing optimal solution in certain situation. Yin [4] required more efficient query optimization techniques specially the cost optimization technique for its query engine. Another issue of document ordering while using XML document is not handled by Park [6].

In addition to the core ideas for integrating XML and OLAP discussed earlier. Wang et al. [7] presented an aggregation operator GXaggregation for XOLAP. On the basis of GXaggregation operator, another Operator XCube is defined which is an extension of tradition cube operation. Both operators can be embedded into XML query language such as XQuery. XOLAP can be used for analysis of XML information in XML warehouse or information integration system. The operators of XOLAP can easily be embedded in existing XML query languages. Likewise, Bordawekar et al. [5] presented structural Group-By, Topological roll-ups and Topological cube operators as new aggregation operators. So

Bordaweker [5] provided more variety by defining multiple aggregation operators than wang[7]. Wiwatwattana et al. [9] presented new definition for cube operator (X^ 3) for XML data which includes a suitable generalized mechanism for specification by providing a new algorithm. Hachica et al. [11] also defined XOLAP operators in TAX tree in three different categories. The categories of these operators are structural operators (Rotate, Switch, Push and Pull), set operators (Slice and Dice) and granularity-related operators (Roll-up, Drill-down and Cube). The approach of operators defined by Hachica [11] is more general and precise than Wiwatwattana [9] as their implemented operators handles the problem of summarizability and heterogeneity in XML tree structure. Wang [7], Bordawekar [5], Wiwatwattana [9] and Hachica [11] worked on similar concepts for defining new operator for efficient XOLAP but Hachica [11] contribution is most amongst them.

Another aspect of integration of XML and OLAP is the construction of OLAP cube over heterogeneous data sources including XML. Park et al. [6] developed XML cube named as XQ-cube, which is efficient in representing XML-OLAP warehouse. Jian et al. [8] presented an IX-cube (Iceberg Cube) over XML data. The IX-Cube and techniques used for computing data cubes from XML data is quite different than computing data cubes on relational data. Farid et al. [10] defined XDXML cubes, XDXML facts and XDXML schemas from XWarehouse data by using their proposed interface XDXML. Boussaid et al. [12] implemented X-Warehousing which stores the XML document and provides an abstraction layer for their analysis. The whole warehouse documents modeled at physical model and represented in form of OLAP cubes. Their major contribution is modeling of complex heterogeneous data in order to build X-warehouse and XML cubes through uniform XML format. Hummer et al. [13] presented XCube [13], an open, manufacturer independent and XML based document template for storing, exchanging and querying the OLAP data. The most important structures of XCube are XCubeSchema, XCubeDimension and XCubeFact. Frank et al. [19] proposed an approach for integrating semantically related data from heterogeneous data sources using XML techniques by transforming all data cube into XML formats in order to integrate data into global virtual cube for sharing the data over the internet by using *XQuery. Transformation of all cubes in to XML formats.* Fong et al. [20] provided the idea of both relational and XML cubes into Relational-XML data warehouse

enabling user to select a familiar OLAP languages. Nappila et al. [21] also implemented a prototype for OLAP cube construction and data integration. Niemi et al. [1], [25] constructed an OLAP cube on demand from distributed heterogeneous data warehouse for analysis requirements which will be smaller in size. They utilize the XML language and provide an XML presentation for OLAP cubes and its schema. Ravat et al. [30] presented a multidimensional OLAP model called Galaxy model for the analysis of XML documents which allows the analysis of XML document and their metadata with the use of numeric and textual indicators [30] and doesn't restrict the decision makers for limited and predefined analysis. Ykhlef [32] proposed a XML data cube model which is a well-founded approach for representing OLAP data by using XML. He showed how OLAP queries on XML data can be expressed in XRQL then an extension in XRQL is presented for expressing more complex OLAP queries. Zhao et al. [33] presented a XML cube model called XMLCube for OLAP on XML data which is based index structure for group by aggregation query over XMLCube.

By making a comparison, Niemi [1], [25], Park [6], Jian [8], Farid [10], Boussaid [12], Hummer [13], Frank [19], Fong [20], Nippila [21] all work on similar concept of computing OLAP cubes from different data sources. They differ in utilizing the data source like Park [6], Farid [10], Jian [8], Boussaid [12] defined their cubes over XML-OLAP warehouses and XWarehouses. Niemi [1], [25] utilizes the distributed heterogeneous data warehouse for computing OLAP cube on demand. Fong [20] computed both XML and Relational cubes but they are restricted to their corresponding languages in order execute OLAP query. No query language or generic query evaluator is proposed that can simultaneously utilizes both types of cubes simultaneously. Frank [19] utilizes the entire heterogeneous data source by transforming all the data cubes into XML format. This will resolves the semantic discrepancies and the integration of data into global cube through XQuery utilizes the integration power of internet. Support of query evaluator is still missing in proposed architecture to utilize the advantage of uniformed format of OLAP cubes. The size of the XML cube is another issue need to be handled. Nippila [21], Jian [8], Ravat [30], Ykhlef [32] and Zhao [33] utilized only XML data for computing OLAP cube over them but support of other data sources is not available like Frank [19] along with heterogeneity issues was not handled. Boussaid [12] not handles the issue of updating XML cube unlike Jian [8]. Niemi

[1] utilized distributed heterogeneous data warehouse for computing cubes but they did not provided the support of XML data source. They focused on the construction of cube on demand and smaller in size for efficient processing of the OLAP queries over them. Niemi [25] also provided the concept of grid technology to distribute the computation which not only reduces the network traffic in distributed environment but also speed up the process of cube construction and OLAP query answering. None of others much contributed on these concepts. The problem of low efficiency of aggregation queries on XML data cube is only solved by Zhao [33] by utilizing index structure. None of other handled this problem. While the support for expressing complex OLAP queries over XML data cube is only provided by Ykhlef [32].

Other core issues arise during the integration of XML and OLAP is handling of dimension hierarchies and correct aggregation. By comparing these issues, the proposed model of Pokorny [16], Frank [19], Pederson [22] and Jensen [27] has provided support to handle the dimension hierarchies while Niemi [1], [25] doesn't provide support to handle the issue of dimension hierarchies. The dimension hierarchies are partly normalized due to not using of snowflake schema. Perderson [22], Wang [7] and Niemi [1], [25] also provided support of correct aggregation at their proposed model while concept of recursive aggregation is only implemented by Wang [7], none of other contributed for implementing the concept of recursive aggregation.

One of the critical problems arises during integration of XML and OLAP is heterogeneity. By comparing the literature reviewed in earlier section, Bruckner et al. [24] identified and resolved the four types of semantic heterogeneity that occurs during integration process. Hummer [13] also handled the semantic heterogeneity problem in their architecture. Frank [19], Pederson [22] and Jian [8] also provided the support of handling heterogeneity. while Frank et al. [21] utilized the uniform format by transforming all the cubes into XML format and resolved the semantic heterogeneity problem while Boussaid et al. [12] also applied the concept of uniform format of XML for creating XML cubes and X-warehousing. Hummer [13] and Nippila [21] provided a standards data exchange format for

heterogeneous data sources for handling the heterogeneity problem but its approach is lacking to completely resolve the heterogeneity problem specially semantic and syntactic heterogeneity. By Implementing the similar concept Bourret et al. [28] applied a utility for transforming the data between XML documents and relational databases, hence avoiding the heterogeneity issue.

Our proposed architecture is motivated from work described in previous section by Pederson [7], Frank [19] and Niemi [1]. We presented a multilayered architecture comprises of three components called storage manager, query manager and client interface. The storage component represents the data and storage layer of the architecture which utilizes the heterogeneous data resources including flat files, operational systems, data warehouses and XML files as Niemi [1], instead of only utilizing XML data source. The ROLAP/MOLAP cubes are being built from these heterogeneous data sources. These ROLAP/MOLAP cubes are transformed into uniformed XOLAP cubes called X-cubes as Frank [19], Hummer [13] and Nippila [21], for automatically resolving the heterogeneity issues by utilizing the uniform format. The capability of storage manager to utilize heterogeneous data sources enables it to compute and store the XOLAP cubes from different domains like sales, expense, disaster management, education etc. The storage manager also maintains the meta-cube which is used to store the necessary information of each cubes stored locally at storage manager or globally at some other location as distributed cubes. The query manager represents the query processing layer which works as a query evaluator similar to the federation manager of Pederson [2]. This component handles the OLAP query requested by the user at client interface which represents the client layer of the proposed architecture. The query manager has ability to handle the OLAP queries belong to different domains by utilizing the meta-cube and executing the query to its appropriate cube for answering. The proposed architecture also provides the support of complex OLAP queries as Ykhlef [32].

Table 2.1 (A) Criteria based Comparison between Integration Techniques of XML and OLAP

- **Criteria1 (XML-OLAP Integration)**
- **Criteria2 (Support for Query Language/ Evaluation/ Optimization/ Translation)**
- **Criteria3 (Support for Heterogeneous Data Sources)**
- **Criteria4 (Handling of Dimension Hierarchies and Correct Aggregation)**
- **Criteria5(Support of Distributed uniformed OLAP cubes to handle heterogeneity)**

Pederson et al. [2]	• Available	• Available	• Available	• Available	• Not Available	• The cost based query optimization techniques improve querying execution and performance.	• Top-down and bottom-up in-lining strategies both fails to find the optimal solutions in certain situations.
Pederson et al. [3]	• Available	• Available	• Available	• Available	• Not Available	• The employment of basic components increases the robustness and reliability of federation • Reduces average retrieval time of data from distributed sources	• several data sources may not be sufficient if the XML document is removed permanently • Problem with consistency of data
Yin et al. [4]	• Available	• Available	• Not Available	• Available	• Not Available	• XML-extended OLAP queries based on physical algebra.	• Required advance query optimization, data loading, query evaluation and cost estimation technique for improved performance
Wang et al. [7]	• Available	• Not Available	• Not Available	• Available	• Not Available	• Recursive aggregation is permitted.	• choice for selecting the adaptive storage structure from multiple storage is not handled
Wiwatwatta na et al.[9]	• Available	• Not available	• Available	• available	• Not available	• Implementation of the summarizability property to get efficiency	• Unable to automated system for determining the cube lattice properties.
Hachica et al. [11]	• Available	• Available	• Not available	• Not mentioned	• Not available	• integration of TAX operators with XOLAP to achieve better performance and efficiency	• limited support for complex analytical queries. • Unable to generate automatic optimal XOLAP queries.

Table 2.1 (B) Criteria based Comparison between Integration Techniques of XML and OLAP

Authors	Criteria1	Criteria2	Criteria3	Criteria4	Criteria5	Features/ Strength	Limitations
Hummer et al. [13]	• Available	• Not available	• Available	• Available	• Not Available for Distributed Cubes	• The semantic heterogeneity problem can be solved by using XCube standards.	• That XML documents tend to be rather large.
Frank et al. [19]	• Available	• Not available	• Available	• Available	• Not Available for Distributed Cubes	• uniform XML formats, will resolves the semantic discrepancies and the integration of data into global cube	• No support for data cubes from different domains. • No Support to answer complex OLAP queries
Nappila et al. [21]	• Available	• Available	• Available	• Not available	• Not Available for Distributed Cubes	• heterogeneous sources are tied under a single format of XML which makes system efficient	• heterogeneity problem not handled • does not provide high level primitives for construction of OLAP cube
Kit et al. [23]	• Available	• Not available	• Not available	• Not mentioned	• Not available	• utilization of topological algorithm will increases the efficiency of the system	• topological operator's enhancement is required • proposed algorithms required partitioning techniques
Jensen et al. [27]	• Available	• Not available	• Not available	• Available	• Not available	• Integration at conceptual level eases the design of OLAP DB's • support of dealing the dimensions hierarchies	• Required efficient query processing techniques such as query translation.
Ravat et al. [30]	• Available	• Not available	• Not available	• Available	• Not available	• allows the analysis of XML document and their metadata with the use of numeric and textual indicators	• not utilized the XML documents with both DTD and XML schemas • Required query optimization techniques for speeding up the processing of XML data
Ykhlef [32]	• Available	• Available	• Not available	• Available	• Not available	• Support for expressing complex OLAP queries	
Zhao et al. [33]	• Available	• Available	• Not available	• Available	• Not available	• solved the problem of low efficiency of aggregation queries over XML cubes	• The problem of controlling redundancy in the index and size of the index not handled.
Proposed Architecture	• Available	• Available	• Available	• Available	• Available	• Supports distributed heterogeneous data source. • Uniformed cubes handles heterogeneity • Support OLAP cubes from different domains • Supports Distributed OLAP queries • Support for complex OLAP queries	• Optimization techniques required for distributed and complex OLAP queries for efficiency.

2.3 Summary

This chapter provides the comprehensive review and critical evaluation of each method or technique used for the integration of XML and OLAP by comparing the previous research contributions. The limitations and strengths of each research contribution are discussed along with providing the solutions of these limitations by proposing a new architecture. The features of our proposed architecture are compared for similarities and differences with others. Finally a comparison between existing and proposed architecture is described in tabular form.

Problem Statement

3. Problem Statement

Past study in area of XOLAP identified many open issues still to be resolved or they need better solutions.

- Pedersen et al. [2] and Niemi et al. [1] research work supports only heterogeneous data sources but no support for uniformed cubes.
- Hummer et al. [13], Frank et al. [19] and Nippila et al. [21] proposed models are able to build uniform cubes but no support for query evaluator is provided.
- Limited support for handling of complex OLAP queries is provided by Yukhlef. [32] and Hachicha et al.[11] but both have only utilized XML data.

Therefore, an architecture is required that can gather data from heterogeneous data sources such as data warehouse, operational systems, XML and flat files and generate uniform OLAP cubes along with a query evaluator for the handling the OLAP queries in better way.

Proposed Architecture of MAUDXC

4. Proposed Architecture of MAUDXC

The objective of this chapter to propose an Architecture of Query Evaluator for Uniformed Distributed XOLAP Cubes that can effectively deal with the factors used to compare the previous research contributions in area of XML and OLAP integration presented in chapter 2. This chapter also describes each component of proposed architecture along with their functionality.

4.1. Literature Review Outcomes

Past study in area of XML and OLAP identified many open issues still to be resolved or they need better solutions. The outcomes of the literature review presented in chapter 2 are as following:

- Utilization of heterogeneous data sources such as flat files, relational, multidimensional and XML data for analysis by constructing uniformed and distributed OLAP cubes on them is required, along with a query evaluator for answering both simple and complex OLAP queries. No previous research contribution completely implemented this concept. Niemi [1], [25] provided support for analyzing distributed and heterogeneous data sources but they don't focus on specifying the OLAP cube in uniform format to remove the heterogeneity issues arise due to heterogeneous data sources. Moreover their proposed model doesn't handles the issue like dimension hierarchies.
- Hummer [13], Frank [19] and Nippila [21] proposed models are able to build uniformed OLAP cubes that handle the heterogeneity problems but none of them worked on dealing the distributed data cubes. The query evaluator is also missing in the proposed model of Hummer [13] and Frank [19] while Nippila [21] approach is lacking to handle the heterogeneity issues.
- Handling of complex OLAP queries is only provided by the Ykhlef [32] while Hachica [11] provided limited support for handling complex OLAP queries. But both only utilized XML data source instead of heterogeneous data sources.
- The concept of handling the sparse domain cubes such as sales cube from manufacturing sector, expense cube from banking sector and financial cube from

disaster management enables the query evaluator to answer the OLAP queries of different domain without any ambiguity.

- Utilization of standardized format allows multiple cube can be merged together to form a global virtual cube for handling complex OLAP queries. This concept is only implemented by the Frank [19]. This idea of uniform cubes integration into global virtual cubes can be further extended in two different ways. Firstly, global virtual cube can be constructed from uniformed sparse domain cubes. Secondly global virtual cube can also be integrated from uniformed distributed cubes. Lastly, global virtual can be integrated from uniform sparse domain and uniform distributed cubes. These extensions will drastically enhance the capability of OLAP systems for answering complex OLAP queries.

4.2 Research Motivation

These outcomes provided the motivation for proposing a novel architecture called Multi-Layered Architecture of Uniformed Distributed XOLAP Cubes (MAUDXC) as shown in Fig. 3.1. MAUDXC has capability to integrate heterogeneous data sources. Integration of heterogeneous data sources usually required a uniform data format. The MAUDXC utilizes the uniform format for integration of heterogeneous data sources along with providing the query evaluator which is capable to answer the simple and complex OLAP queries thus making the architecture more dynamic. MAUDXC is combination of four layers called data layer, storage layer, query processing layer and client layer. These layers are represented by the three different components as described under:

The first component of MAUDXC is called storage manger which represents the data and storage layers capable to utilize the heterogeneous data sources which include the XML data, relational data, warehouse data and flat files. This component constructs and stores the ROLAP/MOLAP cubes depends upon requirements from heterogeneous data sources such as operational systems, text files, XML files and data warehouses. These ROLAP/MOLAP cubes may belong to sparse domains and may be distributed in nature. We have chosen XML as unified representation format due to the analysis over XML data, is more flexible. These ROLAP/MOLAP cubes are then transformed into a single uniform format of XML cubes

called X-Cubes at storage manager. This component also maintains the metadata of these cubes called meta-cube which is used to store all necessary detail such as cube name, its schema name, dimensions and measures name, the database name and the locations of the cubes, if it is stored at any other location instead of storage manager. Therefore meta-cube acts as global data dictionary of distributed database systems for the handling of distributed cubes.

Second component of MAUDXC is Query Manager which represents the query processing layer. Query Manager is the core component comprises of query analyzer, query matcher and query evaluator. Query manger first analysis the OLAP queries through query analyzer to identify the dimensions, measures and cube name. This information and OLAP query is forwarded to query matcher for further processing. Query matcher utilizes the meta-cube of storage manager for identifying the required cube. If information received from query analyzer is matched with any of the record of meta-cube. The required detail which includes the cube name, dimensions name, schema name, database name and location of the cube is passed to query evaluator along with the OLAP query. If no match is found then query matcher notify the error message to user through user interface.

The query evaluator is combination of two sub-component called query modifier and query executer. On the basis of the information received from query matcher, query modifier re-writes the OLAP query. The modified query is then executed on desired cube by the query executer for the result which is forwarded to the client component for displaying it to the requesting user.

The last component of the MAUDXC is client component representing the client layer, which offers a user interface where OLAP query can be provided by the user. The client component forwards the OLAP query to the query manager for the processing of the query.

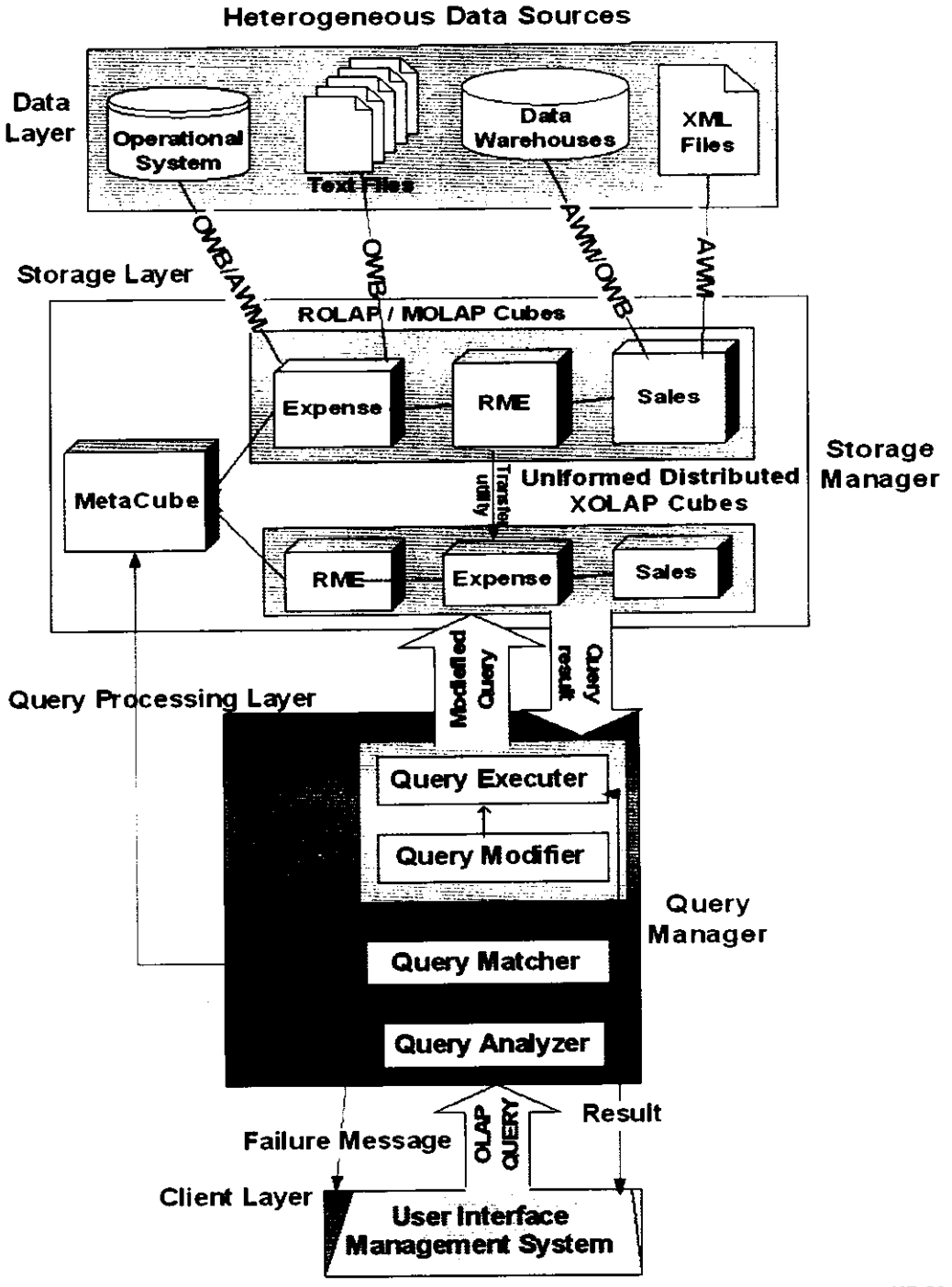


Fig. 4.1 Multi-Layered Architecture of Uniformed Distributed XOLAP Cubes (MAUDXC)

The MAUDXC can also extend the concept of uniformed cube integration into a global virtual cube by providing the global virtual cube integration from sparse domain cubes and distributed cubes. Therefore, complex OLAP queries from sparse domain cubes or distributed cubes can be answered by the query manager.

Components of MAUDXC

Following are the detail of each components of MAUDXC along with their formal definitions.

4.3 Storage Manager

This is the first component of MAUDXC representing the data and storage layers. Storage manager provides the support of heterogeneous data sources. The heterogeneous data sources include pools of flat files, relational, multidimensional and XML data. This component acquires the data from heterogeneous sources. ROLAP/MOLAP cubes will be built from these heterogeneous sources. The data acquisition can be done by using different tools such as analytical workspace manager (awm) and oracle warehouse builder (owb) from any of the data sources. In order to make uniform format these cube are transformed into XML cubes called X-Cubes. This transformation takes place by using a transfer utility developed in C#. The uniform format of X-Cubes solves all types of heterogeneity problems that may arise due to heterogeneous sources.

The storage manager also maintains the metadata of these cube called meta-cube. The meta-cube is almost similar to the warehouse metadata that stores relatively less detail than warehouse metadata. Warehouse metadata usually provides separate descriptions of tables and column within these tables. The description of tables includes table name, type, role, DBMS, location and definition etc while description of columns include column name, order in table, data type, null-able /required, default value, edit rules, definition etc. on the other hand meta-cube is only describing the necessary details such as cube name, schema name from which the cube belongs, the dimensions and measures name, the database name and the location of the distributed cubes if they are stored on any other location.

The location will be null if the cube is locally stored at storage manager. This information is sufficient to meet the requirements of the query manager to answer the OLAP queries belongs to sparse domains cubes or distributed cubes. The concept of meta-cube is more close to global data dictionary (GDD) of distributed systems describing the location of distributed data over the network.

4.3.1 Definition of Meta-cube:

Let one record of a meta-cube R is a 6-tuple (D_i, M_k, c, s, db, l), where $D_i = \{d_1, d_2, \ldots\ldots d_i\}$ is set of i independent dimensions, $M_K = \{m_1, m_2, \ldots\ldots m_k\}$ is set of k measure attributes, $c \in C$ where $C = \{c_1, c_2, \ldots\ldots c_n\}$ is set of n cube names, $s \in S$ where $S = \{s_1, s_2, \ldots\ldots s_p\}$ is set of p schema names, $db \in DB$ where $DB = \{db_1, db_2, \ldots\ldots Db_t\}$ is set of t database names, $l \in L$ where $L = \{l_1, l_2, \ldots\ldots l_v\}$ is set of v locations.

Therefore meta-cube R contains r records starting from $R_1, R_2, \ldots\ldots R_r$, Where $r \geq 1$.

4.3.2 Example 1. Suppose a meta-cube is represented by a relation with six attributes as shown in Table 4.1. This relation act as meta-cubes and stores the relevant information of cubes available at storage manager. This information is utilized by the query manager for answering the OLAP queries. The attribute cube_name stores the name of the cubes either stored at storage manager or any other distributed locations, the dimensions attribute stores the set of dimension names while measures attribute stores the set of measure names used in each cube.

Table 4.1 Meta-cube

Cube_name	Dimensions	Measures	Schema	Database	Location
sales_cube	channel, geography, product, time	sales, quantity	Sales_wh	orcl	null
forecast_cube	geography, product, time	best_fit, linear_regression	Sales_wh	orcl	null
expense_cube	category_dim, time_dim	expense	Eexpense_wh	orcl	site2

The schema attribute stores the name of schema or domain from which the cube belongs. The database attribute stores the name of the databases and location attributes stores the location of the distributed cube. The location will be null if the cube is stored locally.

According to the Table 4.1 the cube sales_cube has dimensions channel, geography, product and time. Sales and quantity are the measures attributes, sales_wh is the schema name from which sales_cube belongs and orcl is the name of database.

The null location indicates that sales_cube is stored locally, while location site2 of expense_cube indicating that this cubes is a distributed cube stored at site2.

4.4 Query Manager

This is the second component of MAUDXC which represents the query processing layer. This component act as a query processor for OLAP queries received from client component. This is the core component and main strength of MAUDXC. The query manager is composed of different components and subcomponents. The three major components of query manager are query analyzer, query matcher and query evaluator.

The query manager first receives the OLAP Query from analyst through user interface at client component and forwards this query to query analyzer for discovering the dimensions, measures and cube name. The query analyzer passes this detail to query matcher for searching the desired cube along with OLAP query.

The query matcher consults the meta-cube of storage manager for inquiring the details provided by the query analyzer. If the details are matched with any of the record of meta-cube, the details such as the name of the required OLAP cube along with its schema name, database name and location of the data is forwarded to query evaluator along with the OLAP query. An error message will be forwarded from query matcher to client component if the details are not matched with any of the record of meta-cube.

The query evaluator is combination of two sub-components called query modifier and query executer. The query modifier utilizes the information received from query matcher to re-write the OLAP query by adding the specific details such as schema name as prefix and database name or location as postfix to the cube and dimensions. The schema name facilities

to target the sparse domain cubes, the database name enables to target the cube locally stored at storage component while location of the cube enables to access the distributed cubes.

This modified OLAP query will be executed by the query executer for result which will be provided to client component for displaying it to the requesting user.

4.5 Components of Query Manager

This section describes the formal definition of the each component and algorithm of the query manager.

4.5.1 Definition of Query Analyzer(QA):

Let OQ is an OLAP query presented by the query manager to the QA; there exist the sets D and M and element c on OQ. D_l is the set of dimensions as discovered by the QA from OQ; M_l is the set of measures as discovered by the QA from O. $c_l \in C$ where $C = \{c_1, c_2 \ldots\ldots c_n\}$ is set of n cube names present at storage component. So a 3-tuple is generated namely d_{in} containing D_l, M_l, c_l i.e. $d_{in} = (D_l, M_l, c_l)$.
OQ and d_{in} are forwarded to next component of the query manager for further processing.

4.5.2 Definition of Query Matcher(QM):

Let $d_{in} = (D_l, M_l, c_l)$ is a set received from QA. OQ is the query received from QA. Meta-cube R contains r records starting from $R_1, R_2 \ldots\ldots R_r$. Where $r \geq 1$. $d_{in} \sqsubseteq R_i (D_i, M_k, c)$ i i.e. $i \geq 1 \leq r$. Required cube is not found at meta-cube and an error message will be forwarded. $d_{in} \in R_i (D_i, M_k, c)$ i i.e. $i \geq 1 \leq r$, there must be at least one record R_i in the meta-cube that contains the d_{in}.
The record R_i returned by the QM and OQ is forwarded to next component of the query manager for processing.

4.5.3 Definition of Query Evaluator (QE):

Let *QMx* is the query modifier of *QE*. Let *QEx* is the query executer of *QE*. R_i is record of meta-cube received from *QM*. *OQ* is the query received from *QM*. Let *MQ* (modified query) which is modified by the *QMx* (Query Modifier) using the specific attributes of R_i. $R_i(s)$ is the schema name used as prefix to the cube and dimensions. $R_i(l)$ is the location used as postfix of the cube and dimensions when location is not null. $R_i(db)$ is the database name used as postfix of the cube and dimensions when location will be null. The *MQ* is executed by the *QEx*. The result is forwarded to client component for presentation.

4.5.4 Algorithm of Query Manager:

This algorithm describes the evaluation and execution of the OLAP query received by the query manager from user interface management system.

Input: OLAP query

Output: query result

Part 1:

Query Analyzer (QA):

1. *Receives the OLAP query from Query manager for evaluation*
2. *Search the OLAP query for the following detail*
   a) *Identify the cube name*
   b) *Identify the dimensions name*
   c) *Identify the measure attributes name*
3. *Forwards the identified detail to the next component of the query manager*

Fig. 4.2 Steps of Query Analyzer

// This algorithm searches the cube name, dimensions and measures name from the given OLAP query

Procedure QueryAnalyzer (OLAPQuery)

1. *For each columns of select clause of query*
2. *Do*

   //Search the column name with group functions

3. *measureVar := measures name*
4. *End loop.*
5. *For each sources of from clause of query*
6. *Do*

   //Search the cube and dimensions name

7. *cubeVar := cube name, dimensionVar := dimensions name*
8. *End loop*
9. *Return OLAPQuery, measureVar, cubeVar, dimensionVar*
10. *End Procedure.*

Fig. 4.3 Algorithm of Query Analyzer

Part 2:

Query Matcher (QM):

1. *Receives the identified detail such as cube name, dimensions name and measures name along with OLAP query from QA for further processing*
2. *Search the identified detail in meta-cube of storage manager*
3. *The QM forwards the additional detail and the OLAP query to the next component of the query manager for evaluation*

Fig. 4.4 Steps of Query Matcher

```
// This algorithm searches the details received from QA from meta-cube
Procedure QueryMatcher (OLAPQuery)
1.    Call Procedure QueryAnalyzer (OLAPQuery)
      //Search the meta-cube for the details return by the procedure QueryAnalyzer
2.    For each record of meta-cube
3.        do
              //Search the cube name, dimensions name and measures name
4.            IF no match found then
5.              Print error message "No such cube exists"
6.              Return;
7.            Else
8.              schemaVar: = schema name, dbVar := database name,
                locVar := location
9.            End IF.
10.   End loop
11.   Return OLAPQuery, schemaVar, dbVar, locVar .
End Procedure.
```

Fig. 4.5 Algorithm of Query Matcher

Part 3:

Query Evaluator (QE)

1. *Receives the OLAP query along with additional detail of cube such as cube name, cube dimensions, cube schema, cube database name and cube location.*
2. *The Query Modifier: a subcomponents of QE performs the following task*
   - *Re-write the OLAP query*
3. *The Query Executer: a subcomponent of QE executes the modified query to the required cube and dimensions*
4. *The result of the query is forwarded to the user interface for displaying it to users.*

Fig. 4.6 Steps of Query Evaluator

```
// This procedure re-write the OLAP query into Modified OLAP query for execution

Procedure QueryModifier (OLAPQuery )

1.   Call Procedure QueryMatcher(OLAPQuery)
2.     do
         //add schema as prefix to the cube and dimensions name
3.       Cubevar := schemaVar + "." + cubeVar
4.       dimensionVar := := schemaVar + "." +dimensionVar
5.       IF location is null then
           //Add database as postfix to the cube and dimensions
6.           cubevar := cubeVar + "@" + dbVar
7.           dimensionVar := := dimesnionVar + "@" + dbVar
8.       Else
           //Add location as postfix to cube and dimensions.
9.           cubevar := cubeVar + "@" + locVar
10.          dimensionVar := := dimesnionVar + "@" + locVar

11.      End If
12.    Return Modified OLAPQuery.
13.  End Procedure.
```

Fig. 4.7 Algorithm of Query Modifier

4.6 Putting the Architecture into work

This section describes that how MAUDXC work with the help of an example of book store company, when focusing on the sales of the book store company there will be three major dimensions: Item, location and time. This example will be represented through a step-wise process representing the flow of data of this architecture.

Step 1:

This step represents the storage manager of the architecture MAUDXC which deals with heterogeneous data sources.

Table 4.2 Item

Category	Type	Product
Books	Computer Science	"Introduction to programming", Deitel & Dietel
		"Introduction to E-commerce", Sams
	Management Science	"Principles of accounting", Martin
		"Principles of Management", Sams
Music	Fast	"Peace of Mind", Maiden
		"Ace of Spades", Motorhead

Table 4.3 Location

Region	Country	City
Asia	Pakistan	Islamabad
		Rawalpindi
		Lahore
Europe	America	New York

Table 4.4 Time

Year	Quarter	Month	Day
2009	Quarter 1	January	01-01-2009
		March	10-03-2009
	Quarter 3	August	15-08-2009
2010	Quarter 2	April	26-04-2010
	Quarter 4	December	31-12-2010

The example is representing the relational data source as an input of the storage manager shown in Table 4.2, 4.3 and 4.4. These relational tables will be converted into specific dimensions as described in Table 4.5, 4.6 and 4.7. The ROLAP cubes are being built from these dimensions. These cubes analyze the multidimensional data through OLAP querying. The fact table will transform into the ROLAP cube named Sales_cube as shown in Table 4.8.

Table 4.5 Item Dimension

Item_key	Category	Type	Product
1	Books	Computer Science	"Introduction to programming", Deitel & Dietel
2	Books	Computer Science	"Introduction to E-commerce", Sams
3	Books	Management Science	"Principles of accounting", Martin
4	Books	Management Science	"Principles of Management", Sams
5	Music	Fast	"Peace of Mind", Maiden
6	Music	Fast	"Ace of Spades", Motorhead

Table 4.6 Location Dimension

Location_ key	Region	Countr y	City
11	Asia	Pakistan	Islamabad
12	Asia	Pakistan	Rawalpind i
22	Asia	Pakistan	Lahore
106	Europe	America	New York

Table 4.7 Time Dimension

Time_ key	Year	Quarter	Month	Day
1	2009	Quarter 1	January	01-01-2009
12	2009	Quarter 1	March	10-03-2009
35	2009	Quarter 3	August	15-08-2009
49	2010	Quarter 2	April	26-04-2010
68	2010	Quarter 4	December	31-12-2010

Table 4.8 Sales_cube

Time_key	Location_key	Item_key	Quantity_sold
1	11	2	1500
1	12	2	2500
1	22	2	5000
12	11	1	1000
12	12	1	1500
35	11	2	2400
1	106	6	1000

The ROLAP cubes are structurally relation as described by the Table 4.8. These ROLAP cubes will be transformed into uniform single format of XML cubes called X-Cube in order to handles the heterogeneity problems that arise due to heterogeneous data sources. The uniformed X-Cubes are described in the following Fig. 4.8.

```
<X-Cube id ="SALES">
    <CELL>
        <Time_key > 1 </Time_key>
        <Location_key> 11 </ Location_key >
        <Item_key> 2 </ Item_key >
        <Qty_sold> 1500 </Qty_sold>
    </CELL>
    <CELL>
        <Time_key > 1 </Time_key>
        <Location_key> 12 </ Location_key >
        <Item_key> 2 </ Item_key >
        <Qty_sold> 2500 </Qty_sold>
    </CELL>
    <CELL>
        <Time_key > 12 </Time_key>
        <Location_key> 11 </ Location_key >
        <Item_key> 1 </ Item_key >
        <Qty_sold> 1000 </Qty_sold>
    </CELL>
    <CELL>
        <Time_key > 35 </Time_key>
        <Location_key> 11 </ Location_key >
        <Item_key> 2 </ Item_key >
        <Qty_sold> 2400 </Qty_sold>
    </CELL>
    <CELL>
        <Time_key > 49 </Time_key>
        <Location_key> 22 </ Location_key >
        <Item_key> 5 </ Item_key >
        <Qty_sold> 8000 </Qty_sold>
    </CELL>
</X-Cube>
```

Fig. 4.8 X-Cube Sales

Step 2:

This step represents the client and query manager component of MAUDXC. Client component provides a user interface management systems (UIMS) for interaction with users while query manager provides the query processor. The user submits his request to the UIMS in form of OLAP query for acquiring analytical information. This query is forwarded to the query manager for desired information. The following Fig 4.9 describes the OLAP query in SQL format for requiring quarterly sales of all types of product in each city.

```
Select      quarter, type, city, sum (quantity_sold)
From        Sales, Location, Time, Item
Where       sales.item_key = Item.item_key
AND         sales.location_key = Location.location_key
AND         sales.time_key = time.time_key
Group by    quarter, type, city;
```

Fig. 4.9 OLAP Query in SQL

The query manager first analyzes the OLAP query through query analyzer for identifying the cube name, dimensions and measures. This information is passed to query matcher which consults the meta-cube of storage component for searching this information. If information is found at meta-cube, then the name of the required cube along with its schema name, database name and location of the data is passed to query evaluator along with the query received from query analyzer.

The query modifier which is subcomponent of query evaluator utilizes this information for query re-write and generates a modified query. The following figure 4.10 describes the modified query after using the schema name bookstore and database name db1.

```
Select      quarter, type, city, sum (quantity_sold)
From        bookstore.sales@db1, bookstore.Location@db1,
            Bookstore.Time@db1, bookstore.Item@db1
Where       sales.item_key = Item.item_key
AND         sales.location_key = Location.location_key
AND         sales.time_key = time.time_key
Group by    quarter, type, city;
```

Fig. 4.10 Modified OLAP Query

The modified query is executed by the query executer which is also the subcomponent of query evaluator for result. The analytical information passed to client component for displaying it to requesting user.

4.7 Implementation Detail of MAUDXC:

This section of the chapter describes the implementation details of MAUDXC. This architecture is implemented in two parts. First part is the implementation of the storage manager representing data and storage layers to generate the uniformed and distributed XOLAP cubes from heterogeneous data sources such as operational systems, text files, xml files and data warehouses. The second part is the implementation of query manager representing query processing layer that has ability to answer the simple and complex OLAP queries from these uniformed and distributed XOLAP cubes along with the user interface management system representing client layer for interaction with the users to receive the OLAP query and display the result.

4.7.1 Storage Manager implementation:

- The XOLAP cube generation is achieved by implementing the storage manager of the MAUDXC.
- Oracle 11g release 1 database is used as storage manager. The storage manager provides the support of heterogeneous data sources such as pools of flat files.

relational, multidimensional and XML data. For acquiring the data from these heterogeneous data sources, additional tools are used.

- The Analytical Workspace Manager (AWM) for Oracle 11g is used to acquire the data from xml files and operational systems to build ROLAP cubes.
- The Oracle Warehouse Builder (OWB) for 11g is used for the data acquisition from flat files, operational system and data warehouses to generate ROLAP cubes. ROLAP cubes are generated due the large size of the cubes instead of MOLAP cubes.
- In order to make uniform format these ROLAP are required to be transformed into XOLAP cubes. We have developed a prototype application in C# with framework 2.0 which acts as transformation utility. This utility converts the ROLAP cubes into uniformed XOLAP cubes called X-Cubes.
- The global data dictionary is also implemented through maintaining meta-cube at each site to support the distributed cubes across the network.

4.7.2 Query Manager and User Interface Management System Implementation

- The implementation of query manager and user interface management system is provided by combining them into a single module developed in C# with framework 2.0.
- This module offers an interface where users can present their OLAP queries along with viewing the result of the query.
- The components of the query manager are implemented as the backend functionality of this module. Therefore all components of query manager such as query analyzer, query matcher, query evaluator which includes query modifier and query executer can be invoked by the user interface for the processing of OLAP query and displaying the desired result to the user.

4.8 Methodology

The design science research methodology is used for this research work. Design science research, by definition, changes the state of the world through the introduction of novel artifacts. Thus, design science researchers are comfortable with alternative world-states. Typical design science research effort proceeds as follows [35].

4.8.1 Awareness of Problem:

An awareness of an interesting problem can come from multiple sources: new developments in industry or in a reference discipline. Reading in an allied discipline may also provide the opportunity for application of new findings to the researcher's field. The output of this phase is a proposal, formal or informal, for a new research effort [35].

4.8.2 Suggestion:

The suggestion phase follows immediately behind the proposal and is intimately connected with it, as the dotted line around Proposal and Tentative Design (the output of the suggestion phase) indicates. Indeed, in any formal proposal for design science research, such as one to be made to the NSF (National Science Foundation) or an industry sponsor, a tentative design and likely the performance of a prototype based on that design would be an integral part of the proposal. Moreover, if after consideration of an interesting problem, a tentative design does not present itself to the researcher, the idea (Proposal) will be set aside. Suggestion is an essentially creative step wherein new functionality is envisioned based on a novel configuration of either existing or new and existing elements. The step has been criticized as introducing non-repeatability into the design science research method; human creativity is still a poorly understood cognitive process. However, the step has necessary analogues in all research methods: for example, in positivist research, creativity is inherent in the leap from curiosity about organizational phenomena to the development of appropriate constructs that operationalize the phenomena and an appropriate research design for their measurement [35].

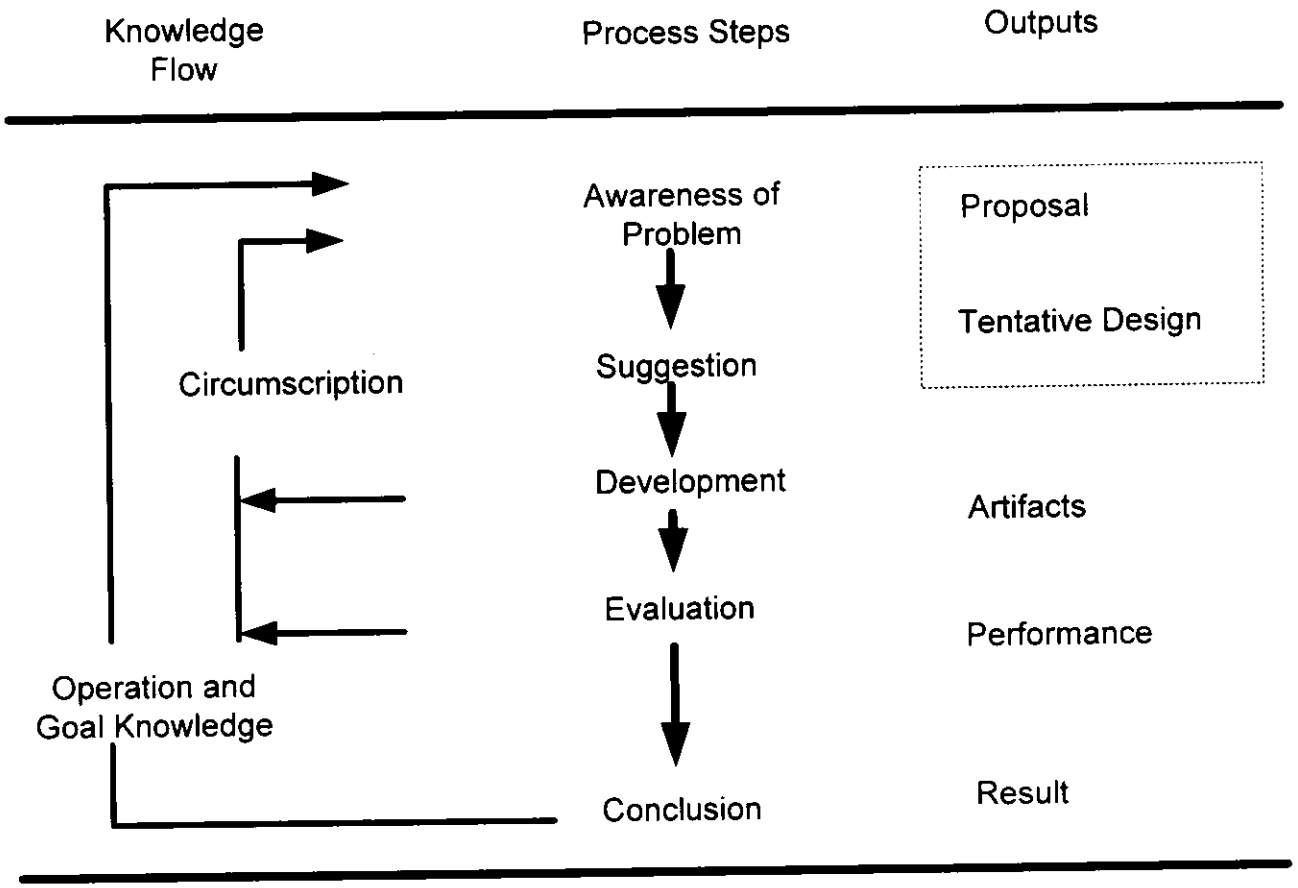


Fig. 4.11: The general methodology of design science research [35].

4.8.3 Development:

The tentative Design is further developed and implemented in this phase. Elaboration of the tentative Design into complete design requires creative effort. The techniques for implementation will of course vary, depending on the artifact to be constructed. An algorithm may require construction of a formal proof. An expert system embodying novel assumptions about human cognition in an area of interest will require software development, probably using a high-level package or tool. The implementation itself can be very pedestrian and need

not involve novelty beyond the state-of-practice for the given artifact; the novelty is primarily in the design, not the construction of the artifact [35].

4.8.4 Evaluation:

Once constructed, the artifact is evaluated according to criteria that are always implicit and frequently made explicit in the Proposal (Awareness of Problem phase). Deviations from expectations, both quantitative and qualitative, are carefully noted and must *be tentatively explained.* That is, the evaluation phase contains an analytic sub-phase in which hypotheses are made about the behavior of the artifact. This phase exposes an epistemic fluidity that is in stark contrast to a strict interpretation of the positivist stance. At an equivalent point in positivist research, analysis either confirms or contradicts a hypothesis. Essentially, save for some consideration of future work as may be indicated by experimental results, the research effort is finished. For the design science researcher, by contrast, things are just getting interesting. Rarely, in design science research, are initial hypotheses concerning behavior completely borne out. Instead, the evaluation phase results and additional information gained in the construction and running of the artifact are brought together and fed back to another round of Suggestion. The explanatory hypotheses, which are quite broad, are rarely discarded; rather, they are modified to be in accord with the new observations. This suggests a new design, frequently preceded by new library research in directions suggested by deviations from theoretical performance. (Design science researchers seem to share Allen Newell's concept [from cognitive science] of theories as complex, robust nomological networks.) This concept has been observed by philosophers of science in many communities (Lakatos, 1978); and working from it, Newell suggests that theories are not like clay pigeons, to be blasted to bits with the Popperian shotgun of falsification. Rather, they should be treated like doctoral students. One corrects them when they err, and is hopeful they can amend their flawed behavior and go on to be evermore useful and productive (Newell, 1990) [35].

4.8.5 Conclusion

This phase is the finale of a specific research effort. Typically, it is the result of satisfying; that is, although there are still deviations in the behavior of the artifact from the (multiply) revised hypothetical predictions, the results are adjudged "good enough." Not only are the results of the effort consolidated and "written up" at this phase, but the knowledge gained in the effort is frequently categorized as either "firm" facts that have been learned and can be repeatedly applied or behavior that can be repeatedly invoked or as "loose end" anomalous behavior that defies explanation and may well serve as subject of further research [35].

4.9 Summary:

This chapter provides the detailed overview of the outcomes of the literature review presented in chapter 2 and describes the proposed architecture called "MAUDXC" which effectively deals with the major challenges such as heterogeneity issues, dimension hierarchies, correct aggregation of data and answering of complex OLAP queries. These issues are common when heterogeneous data sources are integrated to construct OLAP cubes. The MAUDXC is capable to handles these issues along with providing some additional features such as uniformed distributed and sparse domain cubes and their integration into global virtual cube. The MAUDXC utilizes the heterogeneous data source and builds uniformed cubes. The uniform format of cube enables this architecture to handle the heterogeneity issues along with dimension hierarchies and correct aggregation. The uniformed cube may be distributed in nature and belongs to sparse domains. MAUDXC is providing a query manager that utilizes these uniformed distributed and sparse domain cubes to answer the simple and complex OLAP queries.

The significance of MAUDXC is its combination of multiple layers represented by the three major components: storage manager, query manager and client interface which makes MAUDXC a unique architecture. Storage manager component to utilize

heterogeneous data sources for creating and storing uniformed cubes along with meta-cube which stores the description of all cubes stored locally or globally, query manager component to answer the OLAP queries and the client component for interaction with users.

Therefore, MAUDXC provides many research contributions as follows:

- Integration of heterogeneous data sources by using a uniform format
- Concept of uniformed distributed and sparse domain OLAP cubes.
- Integration of these cubes into single global virtual cube
- The query manager for answering of simple and complex OLAP queries by utilizing these cubes.

This architecture opens further research areas in MAUDXC such as extension of query analyzer to analyze OLAP query in any format, extension in query evaluator for query re-write if query analyzer is extended, use of optimization techniques for efficient query processing and more efficient integration of global virtual cube from uniformed distributed or sparse domain cubes.

Scope

5. Scope

The scope of this research work is described as under:

- To generate the ROLAP cube from heterogeneous data sources
- Transform this ROLAP cubes into a uniformed XOLAP cubes called X-Cubes.
- The implementation of meta-cube for representing the description of sparse domain and distributed cubes.
- The implementation of query evaluator that utilizes the meta-cube for OLAP queries execution to appropriate cubes for the result.
- The implementation of interface where user can post OLAP Query and view the result of the query

Validation and Evaluation of the Architecture "MAUDXC"

6. Validation and Evaluation of the Architecture "MAUDXC"

The objective of this chapter includes to

- Test the proposed architecture "MAUDXC"
- Describe and Evaluate the working of proposed architecture "MAUDXC"
- Performance of the proposed architecture "MAUDXC"

These objectives have been achieved by using various datasets of different domains. The MAUDXC is capable of dealing with heterogeneous data sources. These heterogeneous data sources are:

- Operational system
- Text files
- XML structured files

A number of experiments which involves the generation of ROLAP cubes, transformation of the ROLAP cubes into uniformed XOLAP cubes called X-Cubes through storage manager and execution of the OLAP queries on these cubes through query manager of the proposed architecture "MAUDXC".

These experiments have been performed on the data from the above mentioned sources. The details of each of these experiments are given below.

6.1 Experiment 1:

ERRA (www.erra.pk) is an organization which is responsible to manage the post earthquake disaster management activities. ERRA dataset has been used in this experiment. This dataset summarize the number of beneficiaries in various tranches of payment to the different earthquake affected districts of AJ & K and KPK.

The MAUDXC architecture is implemented in two parts.

a) Uniformed cube generation through storage manager
b) OLAP query evaluation through query manager

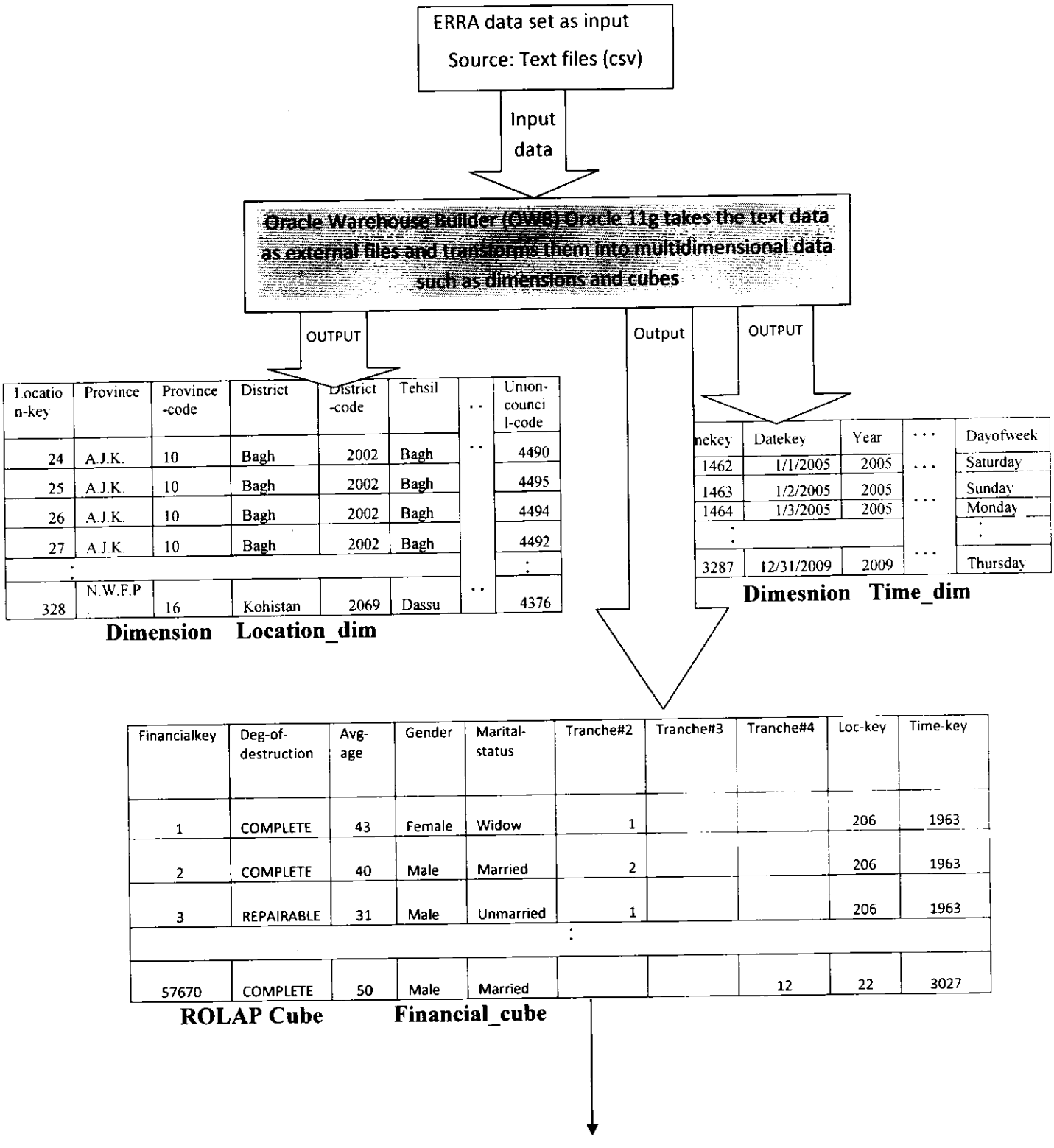
These two steps are described in details visually as well as textually as below.

6.1.1 Uniformed Cube Generation:

The data set used for this experiment is subset of data from ERRA. Earthquake Reconstruction and Rehabilitation Authority (ERRA) is an organization established in 2005 to deal with 2005 earthquake in Pakistan. The data set belongs to disaster management domain which contains the data about locations of earthquake affected areas and the payment records that are made to affected persons in installments according to the level of destruction of their properties.

The data set was in form of text file, converted into MS Excel comma separated vales (CSV) files passes to the Oracle Warehouse Builder (OWB) as input. OWB transformed the data into multidimensional data by creating appropriate dimensions and cubes. In this experiment ROLAP cubes are created instead of MOLAP cubes (due to the large amount of data in cubes). Fig. 6.1 represents the dimensions and the ROLAP cube. Fig. 6.2 represents the transformed X-Cubes.

This process is graphically represented as below:

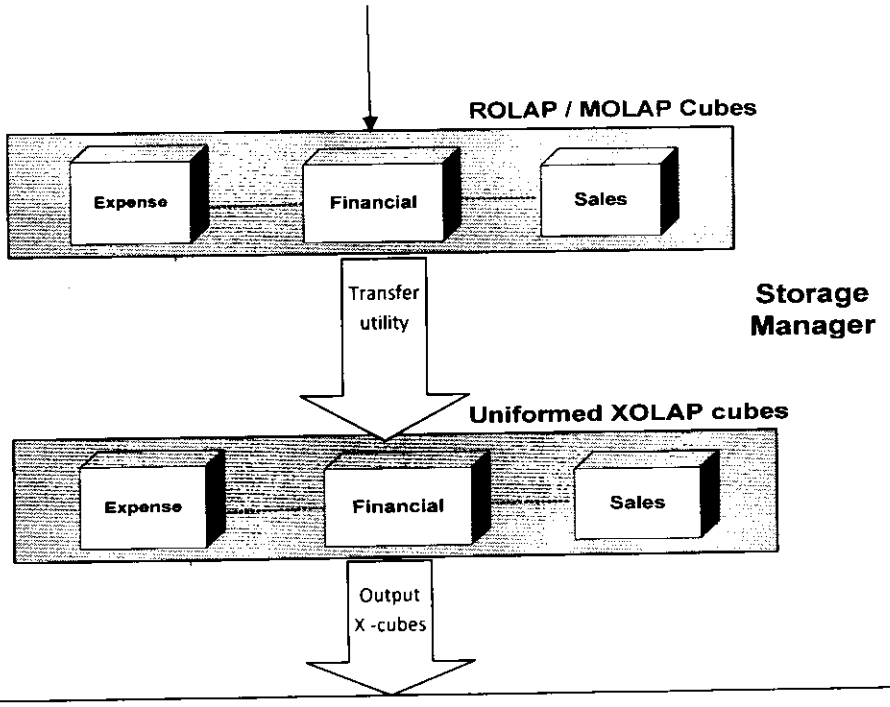


Locatio n-key	Province	Province -code	District	District -code	Tehsil	..	Union-counci l-code
24	A.J.K.	10	Bagh	2002	Bagh	..	4490
25	A.J.K.	10	Bagh	2002	Bagh		4495
26	A.J.K.	10	Bagh	2002	Bagh		4494
27	A.J.K.	10	Bagh	2002	Bagh		4492
:							:
328	N.W.F.P	16	Kohistan	2069	Dassu	..	4376

nekey	Datekey	Year	...	Dayofweek
1462	1/1/2005	2005	...	Saturday
1463	1/2/2005	2005	...	Sunday
1464	1/3/2005	2005		Monday
:				:
3287	12/31/2009	2009	...	Thursday

Financialkey	Deg-of-destruction	Avg-age	Gender	Marital-status	Tranche#2	Tranche#3	Tranche#4	Loc-key	Time-key
1	COMPLETE	43	Female	Widow	1			206	1963
2	COMPLETE	40	Male	Married	2			206	1963
3	REPAIRABLE	31	Male	Unmarried	1			206	1963
:									
57670	COMPLETE	50	Male	Married			12	22	3027

Fig. 6.1 Phase of Storage Manager with Input/Output




```
<X-cube id ="Financial_cube">
        <CELL>
                <Financialkey > 1 </ Financialkey >
                <Deg_of_destruction > complete  </ Deg_of_destruction >
                <avg_age> 43 </ avg_age >
                <Gender> female </ Gender >
                <Marital_status > widow  </ Marital_status >
                <Tranche#2> 1 </ Tranche#2>
                < Tranche#3>  </ Tranche#3>
                < Tranche#4>  </ Tranche#4>
                <Locdimkey> 206 </ Locdimkey >
                <Timedimkey> 1963 </ Timedimkey >
        </CELL>
                .
                .
        <CELL>
                <Financialkey > 57670 </ Financialkey >
                <Deg_of_destruction > complete  </ Deg_of_destruction >
                <avg_age> 50 </ avg_age >
                <Gender> male </ Gender >
                <Marital_status > married  </ Marital_status >
                <Tranche#2>  </ Tranche#2>
                < Tranche#3>  </ Tranche#3>
                < Tranche#4> 12  </ Tranche#4>
                <Locdimkey> 22 </ Locdimkey >
                <Timedimkey> 3027 </ Timedimkey >
         </CELL>
</X-cube>
```

Fig. 6.2 X-Cube Financial_cube

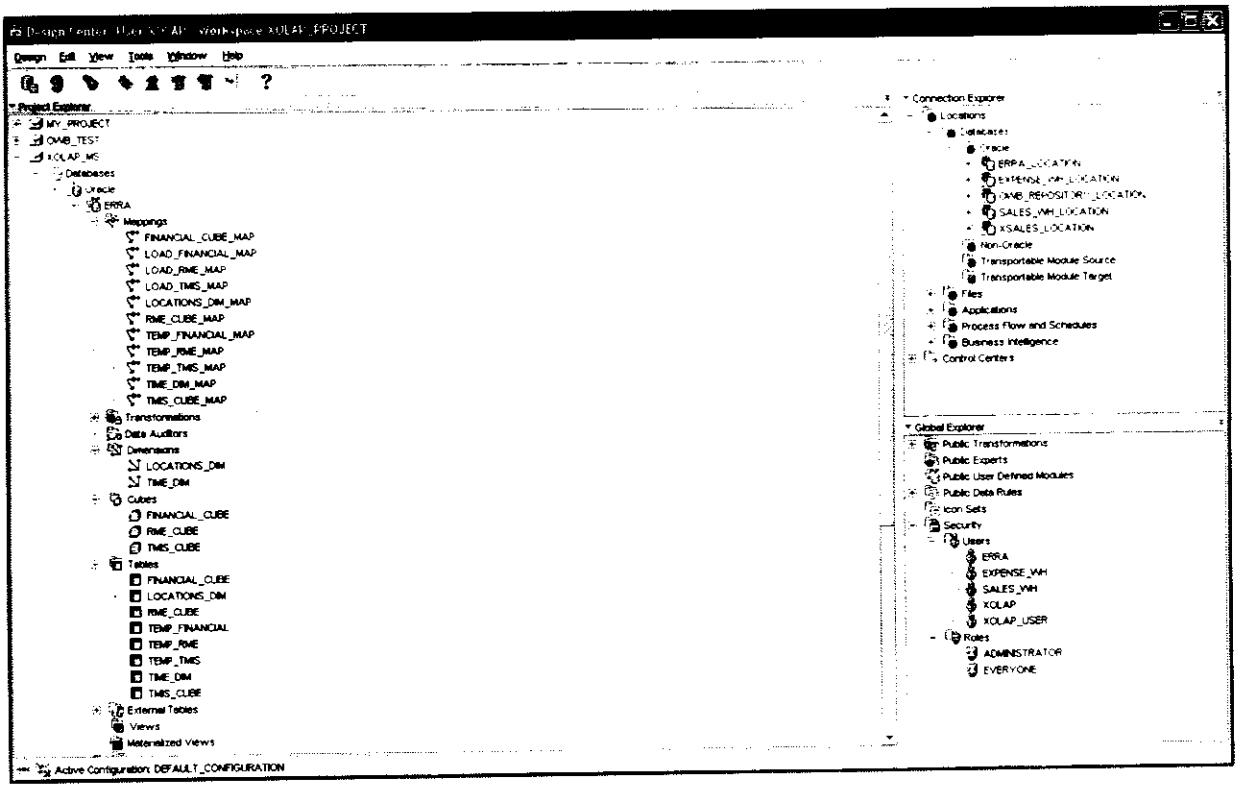


Fig. 6.3 View of OWB for ROLAP cubes and dimension by using ERRA data set

The ROLAP cubes are transformed into uniform XML cubes called X-Cubes by using a transfer utility application developed in C# framework 2.0. Fig 6.4 represents the screenshot of application that converts the ROLAP cube in to uniform XML cube.

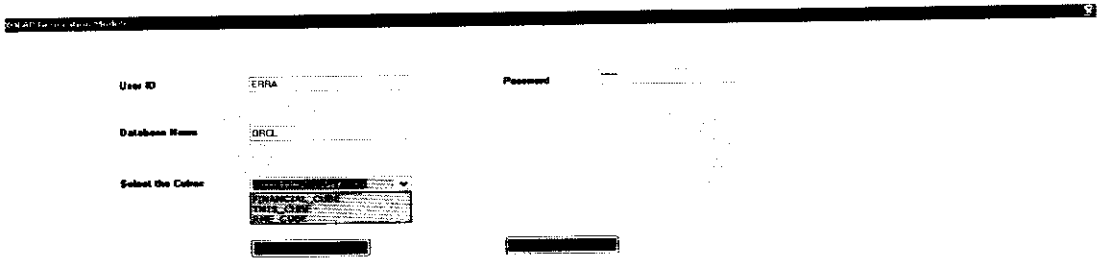


Fig. 6.4 View of ROLAP cubes into XOLAP transformation Interface

6.1.2 OLAP Query Evaluation

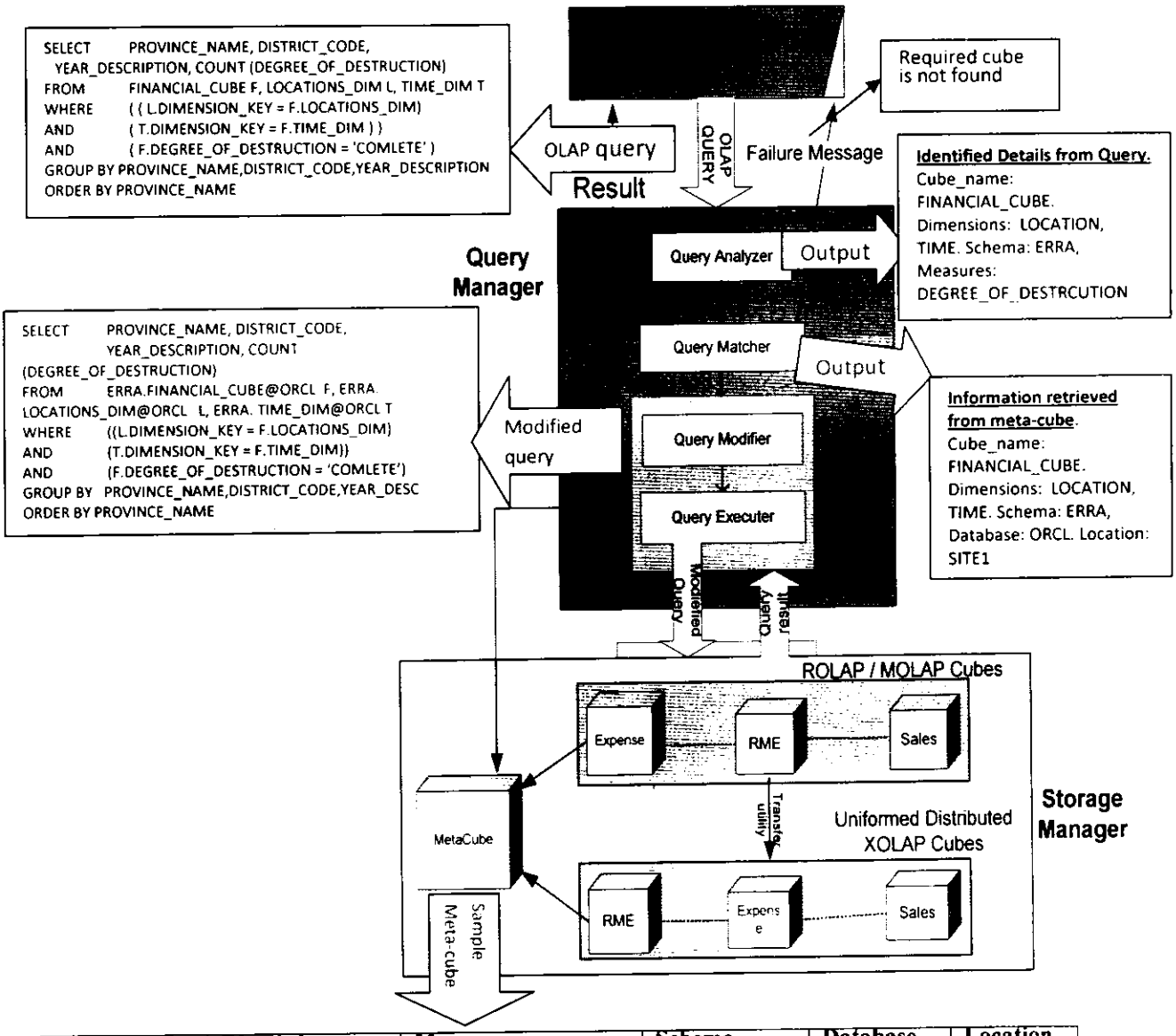
This section graphically represents the query evaluation process of OLAP query posted by the user at client component through user interface management system. The functionality of Query Manager is outlined below:

The OLAP query is posted by the user at UIMS. The UIMS forwarded this query to query manager for the processing of the query as shown in Fig. 6.6. The sample OLAP query is given in Fig. 6.5. The query manager forwards the OLAP query to query analyzer for discovering the dimensions, measures and cube name. The output of the query analyzer is showed in the diagram Fig. 6.5.

The query analyzer passes this detail to query matcher for searching the desired cube along with OLAP query. The query matcher consults the meta-cube of storage manager for inquiring the details provided by the query analyzer. The sample of meta-cube is provided in Fig. 6.5. If the details are matched with any of the record of meta-cube, the details such as the name of the required OLAP cube along with its dimension names, schema name, database name and location of the data is forwarded to query evaluator along with the OLAP query.

The output of the query matcher is also given in the Fig. 6.5. An error message will be forwarded from query matcher to client component if the details are not matched with any of the record of meta-cube. The output of the sample error message is also presented in Fig. 6.5.

The query evaluator is combination of two sub-components called query modifier and query executer



Cube_name	Dimensions	Measures	Schema	Database	Location
Sales_cube	channel, geography, product, time	sales, quantity	Olaptrain	orcl	null
forecast_cube	geography, product, time	best_fit, linear_regression	Sales_wh	orcl	null
expense_cube	category_dim, time_dim	expense	expense_wh	orcl	site2
Financial_cub e	Location, time	Tranche#2, Deg_of_destrcution, Tranche#3, Tranche#4	ERRA	orcl	null

Fig. 6.5 Phases of Query Evaluation Module with Input and Output

The query modifier utilizes the information received from query matcher to re-write the OLAP query by adding the specific details such as schema name as prefix and database name or location as postfix to the cube and dimensions. The schema name facilities to target the sparse domain cubes, the database name enables to target the cube locally stored at storage component while location of the cube enables to access the distributed cubes stored globally. The output of query modifier is provided in the Fig. 6.5.

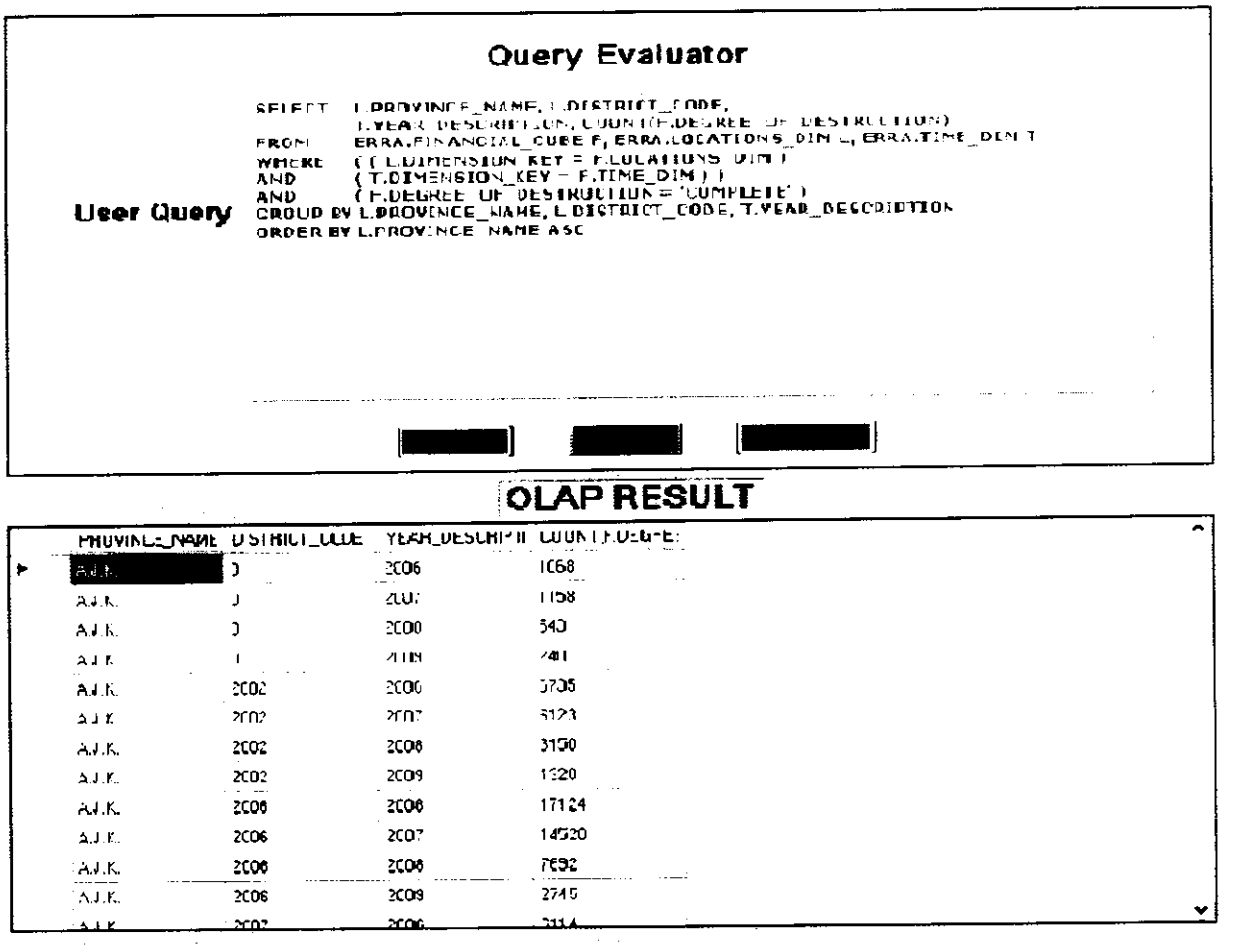


Fig. 6.6 View of Query Evaluator and User Interface Module with OLAP query to ERRA

This modified OLAP query will be executed by the query executer for result which will be provided to client component for displaying it to the requesting user. The result of the OLAP query is shown in the Fig 6.6.

6.2 Experiment 2:

OLAPTRAIN dataset is used for performing this experiment. OLAPTRAIN is a test data which belongs to the sales domain.

The MAUDXC architecture is implemented in two parts.

c) Uniformed cube generation through storage manager
d) OLAP query evaluation through query manager

These two steps are described in details visually as well as textually below.

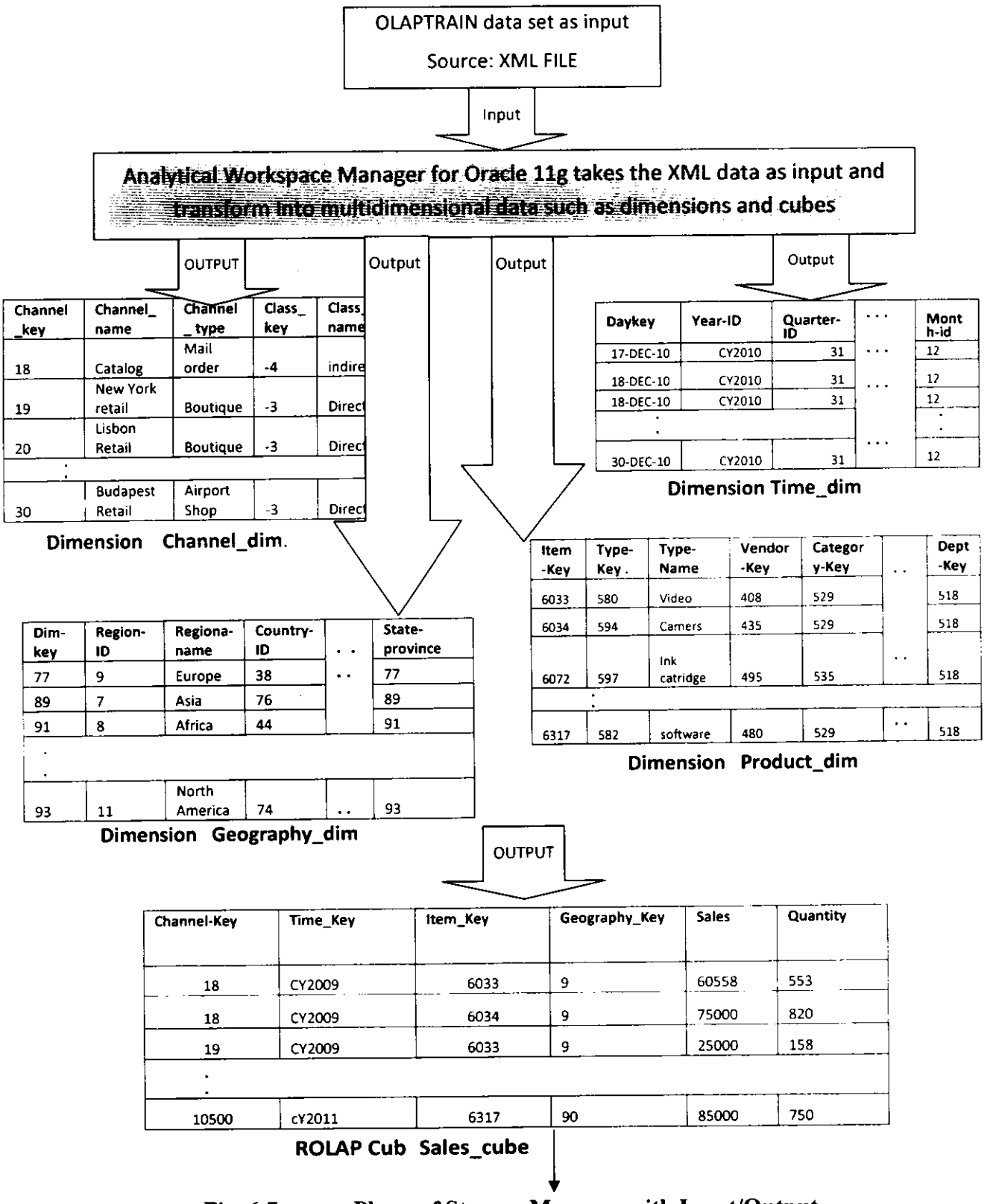
6.2.1 Uniformed Cube Generation:

The data set used for this experiment is subset of data from OLAPTRAIN. The data set belongs to sales domain which contains the data and information about total sales and quantity sold of the products of different categories in different regions over the different time span.

The data set was in form of XML files that are passed to the AWM as input. AWM transformed the data into multidimensional data by creating appropriate dimensions and cubes. The ROLAP cubes are created instead of MOLAP cubes in this experiment due to large cube size. Fig 6.7 represents the dimensions and ROLAP cube while Fig. 6.8 represents the transformed X-Cubes.

The ROLAP cubes are transformed into uniform XML cubes called X-Cubes by using a transfer utility application developed in C# framework 2.0. The screenshot of application that converts the ROLAP cube in to uniform XML cube is described in Fig. 6.4.

This section graphically represents the cube generation process



Channel_key	Channel_name	Channel_type	Class_key	Class_name
18	Catalog	Mail order	-4	indire
19	New York retail	Boutique	-3	Direct
20	Lisbon Retail	Boutique	-3	Direct
:				
30	Budapest Retail	Airport Shop	-3	Direct

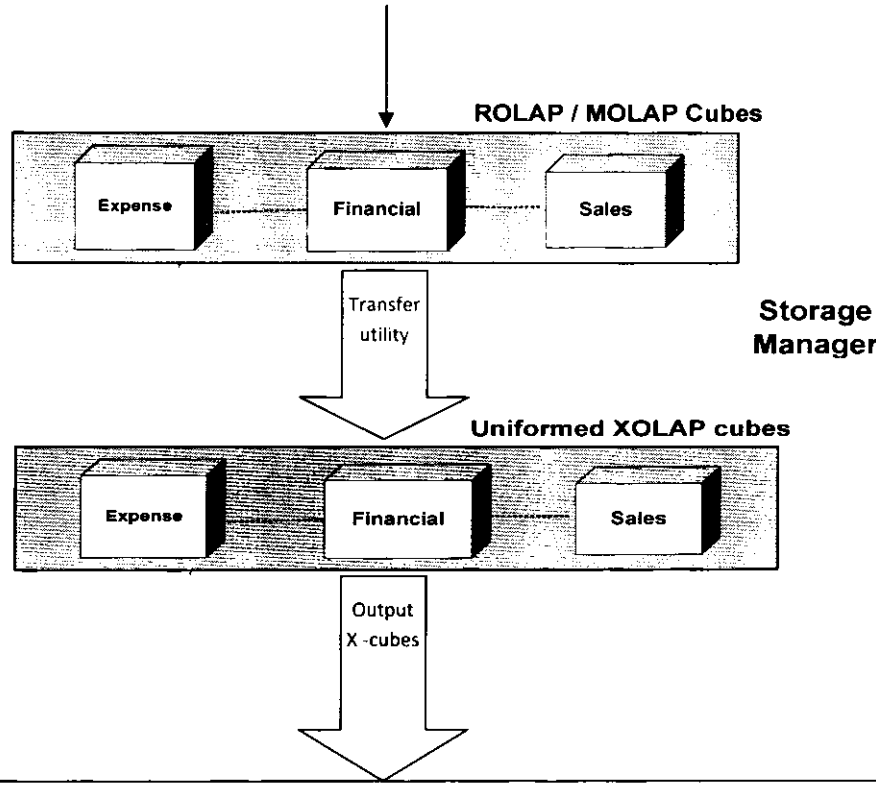
Daykey	Year-ID	Quarter-ID	...	Month-id
17-DEC-10	CY2010	31	...	12
18-DEC-10	CY2010	31	...	12
18-DEC-10	CY2010	31		12
.				.
30-DEC-10	CY2010	31	...	12

Item-Key	Type-Key.	Type-Name	Vendor-Key	Category-Key	..	Dept-Key
6033	580	Video	408	529		518
6034	594	Camers	435	529		518
6072	597	Ink catridge	495	535	..	518
:						
6317	582	software	480	529	..	518

Dim-key	Region-ID	Regiona-name	Country-ID	..	State-province
77	9	Europe	38	..	77
89	7	Asia	76		89
91	8	Africa	44		91
.					
93	11	North America	74	..	93

Channel-Key	Time_Key	Item_Key	Geography_Key	Sales	Quantity
18	CY2009	6033	9	60558	553
18	CY2009	6034	9	75000	820
19	CY2009	6033	9	25000	158
.					
10500	cY2011	6317	90	85000	750

Fig. 6.7 Phase of Storage Manager with Input/Output




```
<X-cube id ="Sales_cube">
        <CELL>
                <ChannelKey > 18 </ ChannelKey >
                <TimeKey > CY2009  </ TimeKey >
                <ItemKey> 6033 </ ItemKey >
                <GeographyKey> 9 </ GeographyKey >
                <Sales > 60558  </ Sales >
                <Quantity> 555 </ Quantity 2>

        </CELL>
                .
                .
        <CELL>
                <ChannelKey > 10500 </ ChannelKey >
                <TimeKey > CY2011  </ TimeKey >
                <ItemKey> 6317 </ ItemKey >
                <GeographyKey> 90 </ GeographyKey >
                <Sales > 85000  </ Sales >
                <Quantity> 750 </ Quantity 2>
                .
        </CELL>
</X-cube>
```

Fig. 6.8 X-Cube Sales_cube

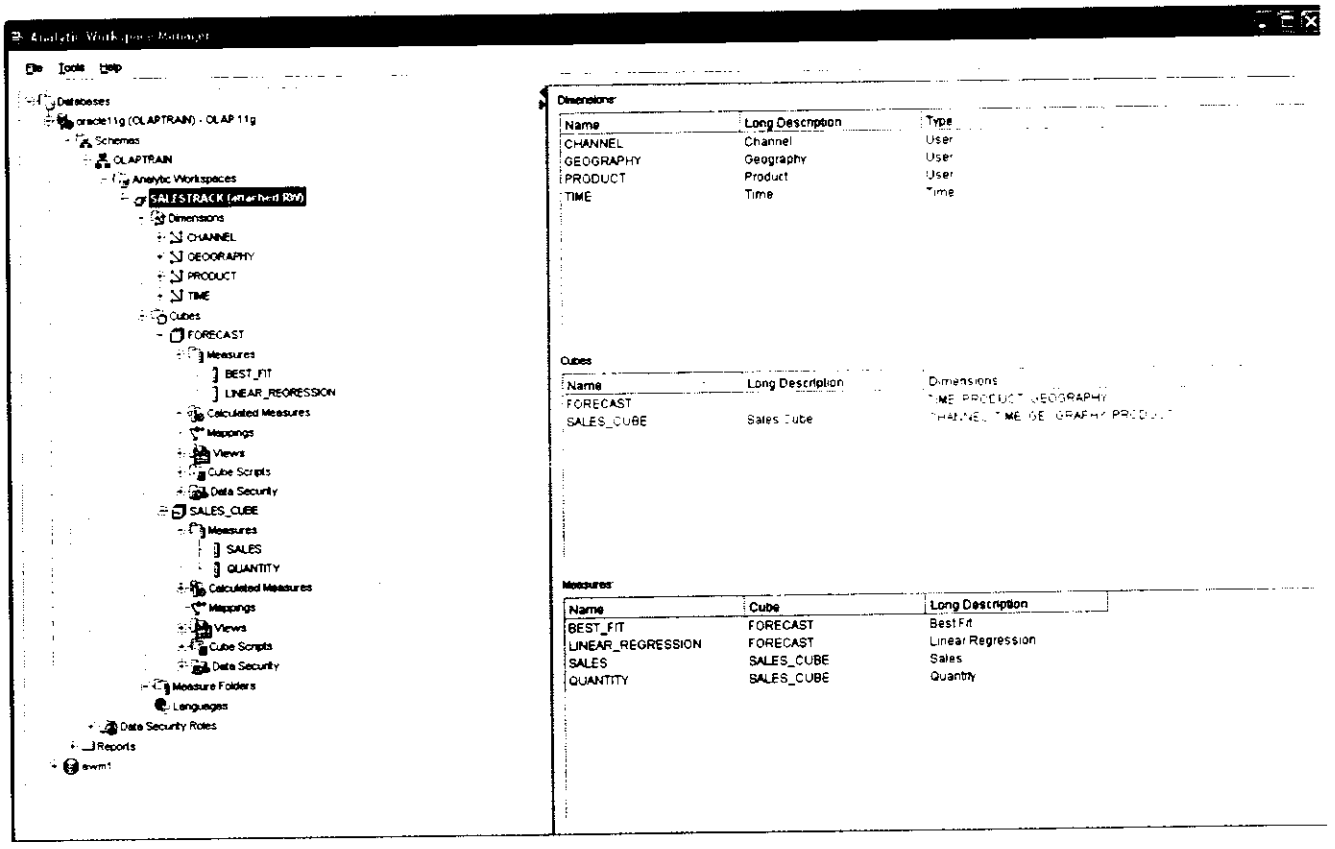
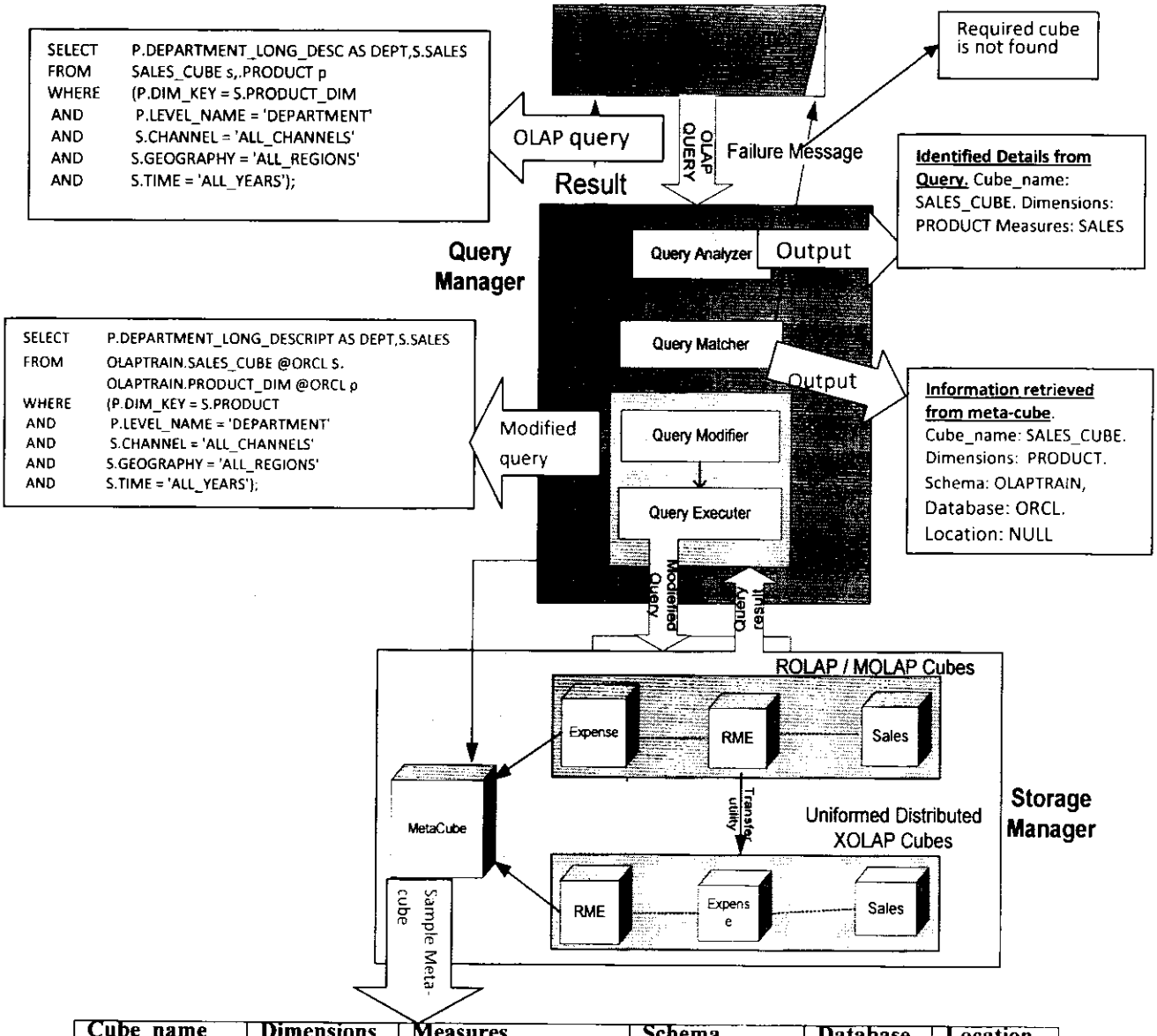


Fig. 6.9 View of AWM for ROLAP cubes and dimension by using OLAPTRAIN data set

6.2.2 OLAP Query Evaluation

This section graphically represents the query evaluation process of OLAP query posted by the user at client component through user interface management system.

The OLAP query is posted by the user at UIMS. The UIMS forwarded this query to query manager for the processing of the query as shown in Fig. 6.10. The sample OLAP query is given in Fig. 6.10. The query manager forwards the OLAP query to query analyzer for discovering the dimensions, measures and cube name. The output of the query analyzer is showed in the Fig. 6.10. The query analyzer passes this detail to query matcher for searching the desired cube along with OLAP query.



Cube_name	Dimensions	Measures	Schema	Database	Location
Sales_cube	channel, geography, product, time	sales, quantity	olaptrain	orcl	null
forecast_cube	geography, product, time	best_fit, linear_regression	olaptrain	orcl	null
expense_cube	category_di m, time_dim	expense	expense_wh	orcl	site2
Financial_cube	Location, time	Tranche#2, Deg_of_destrcution, Tranche#3, Tranche#4	ERRA	orcl	null

Fig. 6.10. Phases of Query evaluation module with input and output

The query matcher consults the meta-cube of storage manager for inquiring the details provided by the query analyzer. The sample of meta-cube is provided in Fig. 6.10. If the details are matched with any of the record of meta-cube, the details such as the name of the required OLAP cube along with its dimension names, schema name, database name and location of the data is forwarded to query evaluator along with the OLAP query. The output of the query matcher is also given Fig. 6.10. An error message will be forwarded from query matcher to client component if the details are not matched with any of the record of meta-cube. The output of the sample error message is also presented in Fig. 6.10.

The query evaluator is combination of two sub-components called query modifier and query executer. The query modifier utilizes the information received from query matcher to re-write the OLAP query by adding the specific details such as schema name as prefix and database name or location as postfix to the cube and dimensions. The schema name facilities to target the sparse domain cubes, the database name enables to target the cube locally stored at storage component while location of the cube enables to access the distributed cubes stored globally. The output of query modifier is provided in the Fig. 6.10.

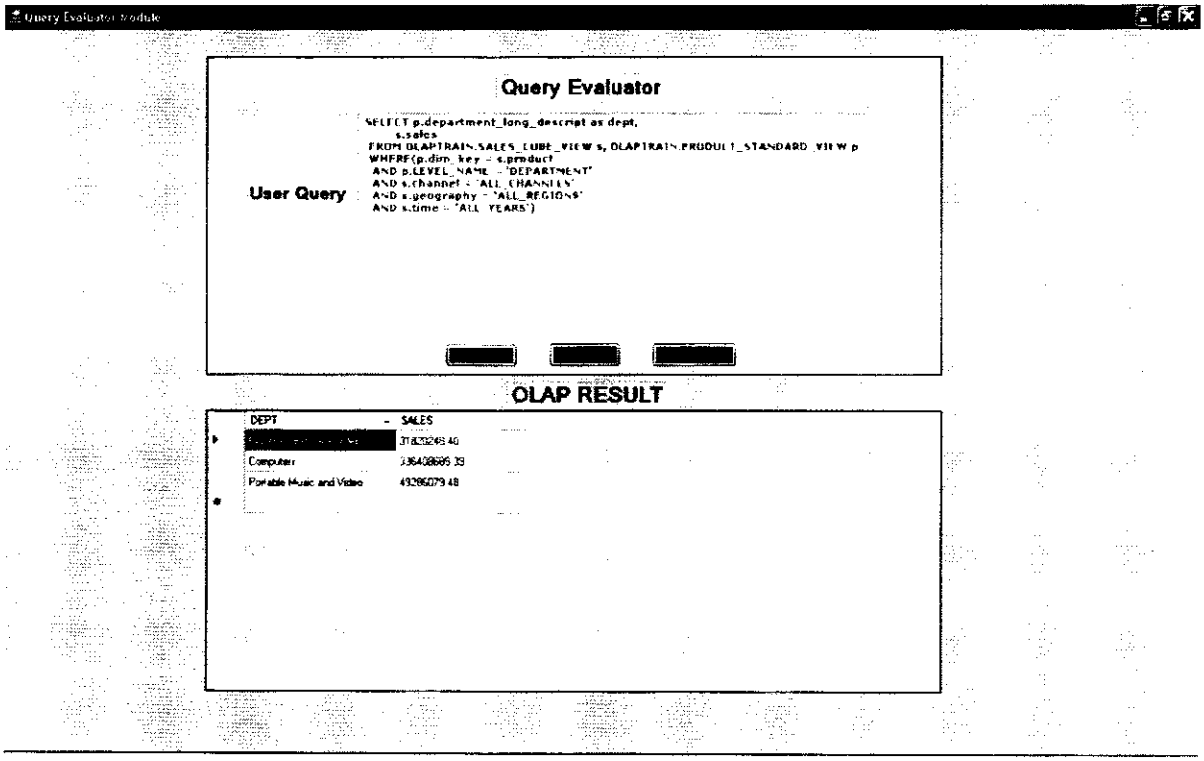


Fig. 6.11 View of Query Evaluator and User Interface Module with OLAP query to OLAPTRAIN

The modified OLAP query will be executed by the query executer for result which will be provided to client component for displaying it to the requesting user. The result of the OLAP query is shown in the following Fig 6.11.

6.3 Experiment 3

Expense_WH dataset is used for performing this experiment. Expense_WH is a test data which represents the expense details.

The MAUDXC architecture is implemented in two parts.

a) Uniformed cube generation through storage manager
b) OLAP query evaluation through query manager

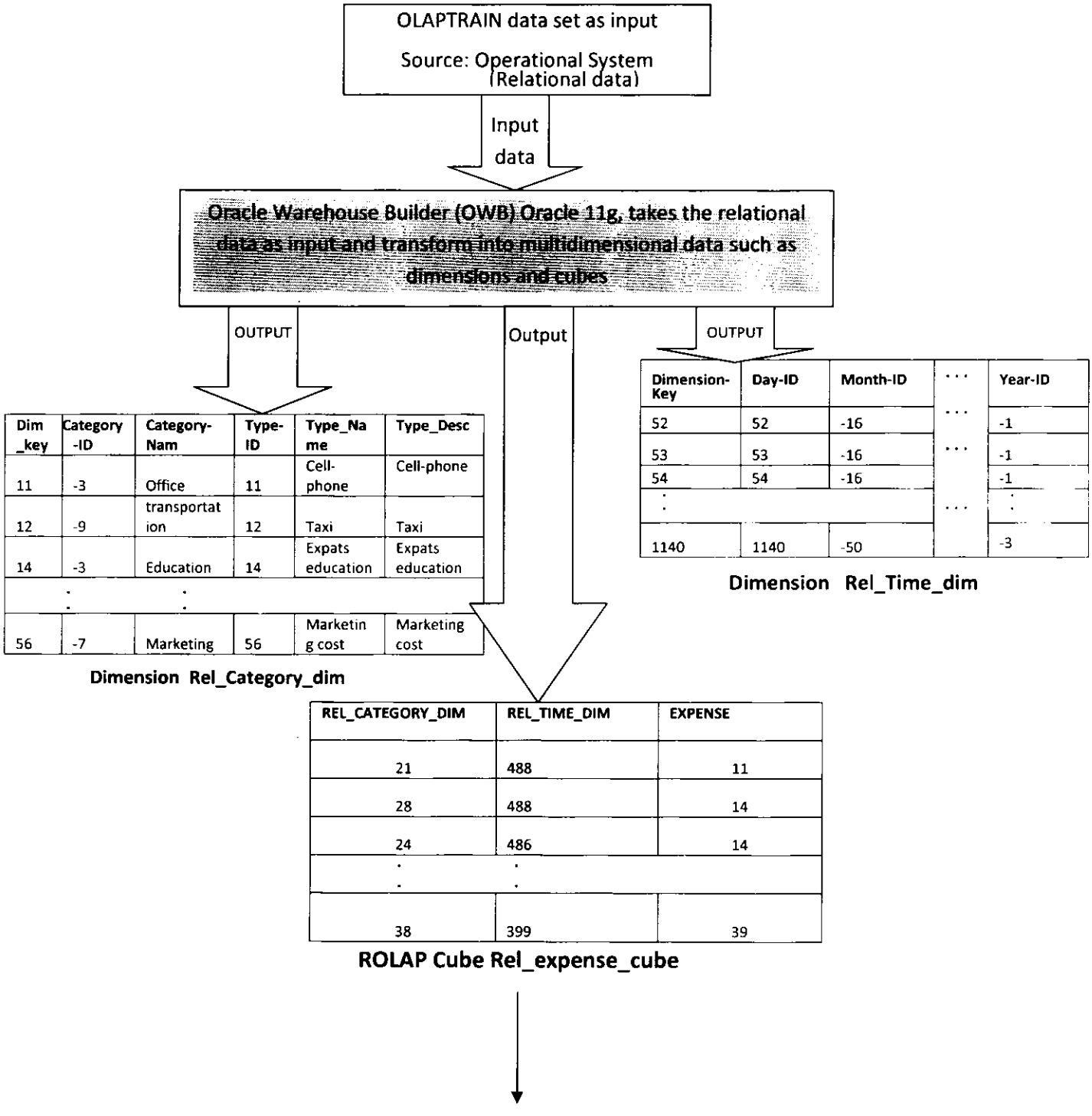
These two steps are described in details visually as well as textually below.

6.3.1 Uniformed Cube Generation:

The data set used for this experiment is subset of data from Expense_WH schema. The data set represents the expense details of an organization.

The data set was in form of relational data that is passed to the OWB as input. OWB transformed the data into multidimensional data by creating appropriate dimensions and cubes. The ROLAP cubes are created instead of MOLAP cubes in this experiment due to large cube size. Fig 6.12 represents the dimensions and the ROLAP cube while Fig. 6.13 represents the transformed X-Cubes.

The ROLAP cubes are transformed into uniform XML cubes called X-Cubes by using a transfer utility application developed in C# framework 2.0. The screenshot of application that converts the ROLAP cube in to uniform XML cube is described in Fig. 6.4.

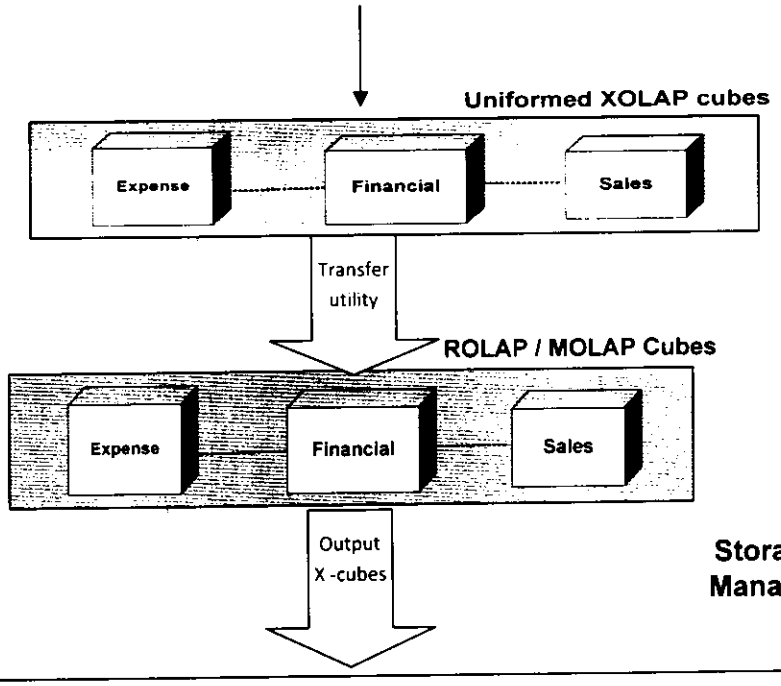


Dim_key	Category-ID	Category-Nam	Type-ID	Type_Name	Type_Desc
11	-3	Office	11	Cell-phone	Cell-phone
12	-9	transportation	12	Taxi	Taxi
14	-3	Education	14	Expats education	Expats education
.	.				
56	-7	Marketing	56	Marketing cost	Marketing cost

Dimension-Key	Day-ID	Month-ID	...	Year-ID
52	52	-16	...	-1
53	53	-16	...	-1
54	54	-16		-1
.			...	.
1140	1140	-50		-3

REL_CATEGORY_DIM	REL_TIME_DIM	EXPENSE
21	488	11
28	488	14
24	486	14
.	.	
38	399	39

Fig. 6.12 Phase of Storage Manager with Input/output




```
<X-cube id ="Rel_Expense_Cube">

        <Rel_Category_Dim > 21 </ Rel_Category_Dim >
        <Rel_Time_Dim > 488  </ Rel_Time_Dim >
        <Expense> 11 </ Expense >

    </CELL>
        .
        .
        .
    <CELL>
        <Rel_Category_Dim > 38 </ Rel_Category_Dim >
        <Rel_Time_Dim > 399  </ Rel_Time_Dim >
        <Expense> 39 </ Expense >
     </CELL>
</X-cube>
```

Fig. 6.13 X-Cube Sales_cube

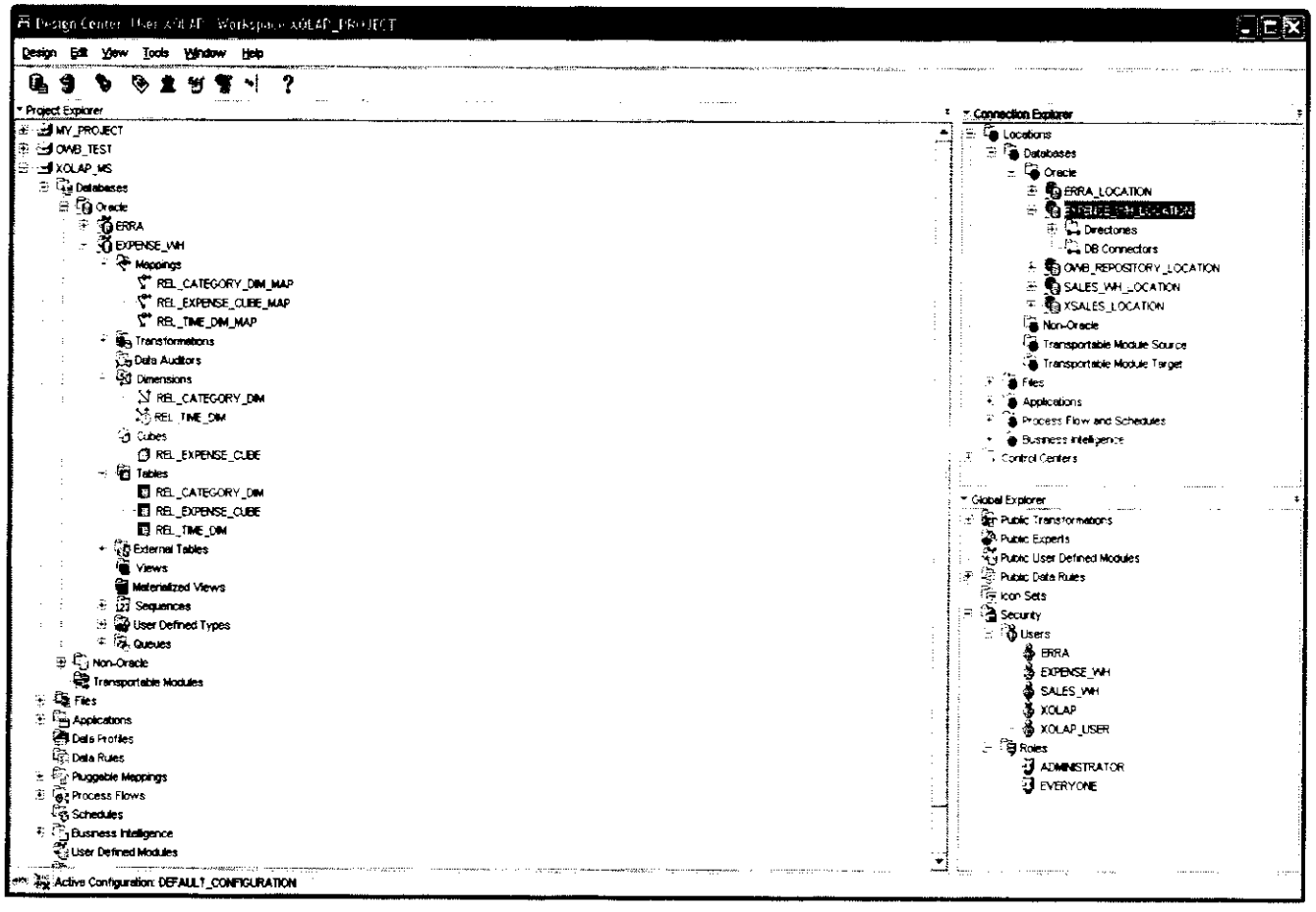


Fig. 6.14 View of OWB for ROLAP cubes and dimension by using Expense_WH data set

6.3.2 OLAP Query Evaluation

This section graphically represents the query evaluation process of OLAP query posted by the user at client component through user interface management system.

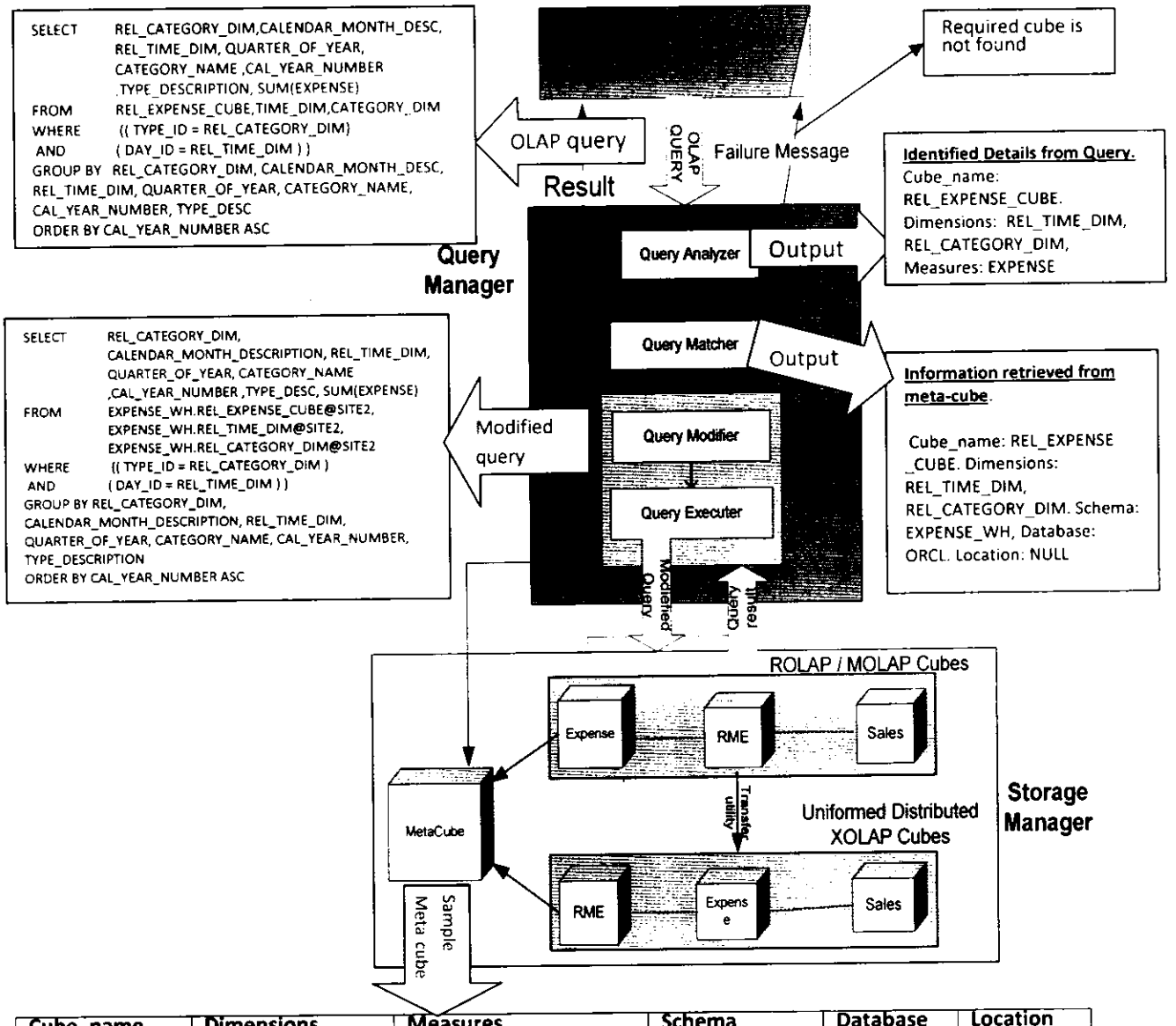
The OLAP query is posted by the user at UIMS. The UIMS forwarded this query to query manager for the processing of the query as shown in Fig. 6.16. The sample OLAP query is given in Fig. 6.15. The query manager forwards the OLAP query to query analyzer for discovering the dimensions, measures and cube name. The output of the query analyzer is showed in the Fig. 6.15. The query analyzer passes this detail to query matcher for searching the desired cube along with OLAP query.

The query matcher consults the meta-cube of storage manager for inquiring the details provided by the query analyzer. The sample of meta-cube is provided in Fig. 6.15. If the details are matched with any of the record of meta-cube, the details such as the name of the required OLAP cube along with its dimension names, schema name, database name and location of the data is forwarded to query evaluator along with the OLAP query.

The output of the query matcher is also given in Fig. 6.15. An error message will be forwarded from query matcher to client component if the details are not matched with any of the record of meta-cube. The output of the sample error message is also presented in Fig. 6.15.

The query evaluator is combination of two sub-components called query modifier and query executer. The query modifier utilizes the information received from query matcher to re-write the OLAP query by adding the specific details such as schema name as prefix and database name or location as postfix to the cube and dimensions. The schema name facilities to target the sparse domain cubes, the database name enables to target the cube locally stored at storage component while location of the cube enables to access the distributed cubes stored globally. The output of query modifier is provided in the Fig. 6.15.

The modified OLAP query will be executed by the query executer for result on a distributed schema Expense_WH which is stored at location site2. The result will be provided to client component for displaying it to the requesting user. The result of the OLAP query is shown in the Fig 6.16.



Cube_name	Dimensions	Measures	Schema	Database	Location
Sales_cube	channel, geography, product, time	sales, quantity	olaptrain	orcl	null
forecast_cube	geography, product, time	best_fit, linear_regression	olaptrain	orcl	null
expense_cube	category_dim, time_dim	expense	expense_wh	orcl	site2
Financial_cube	Location, time	Tranche#2, Deg_of_destrcution, Tranche#3, Tranche#4	ERRA	orcl	null

Fig. 6.15. Phases of Query evaluation module with input and output

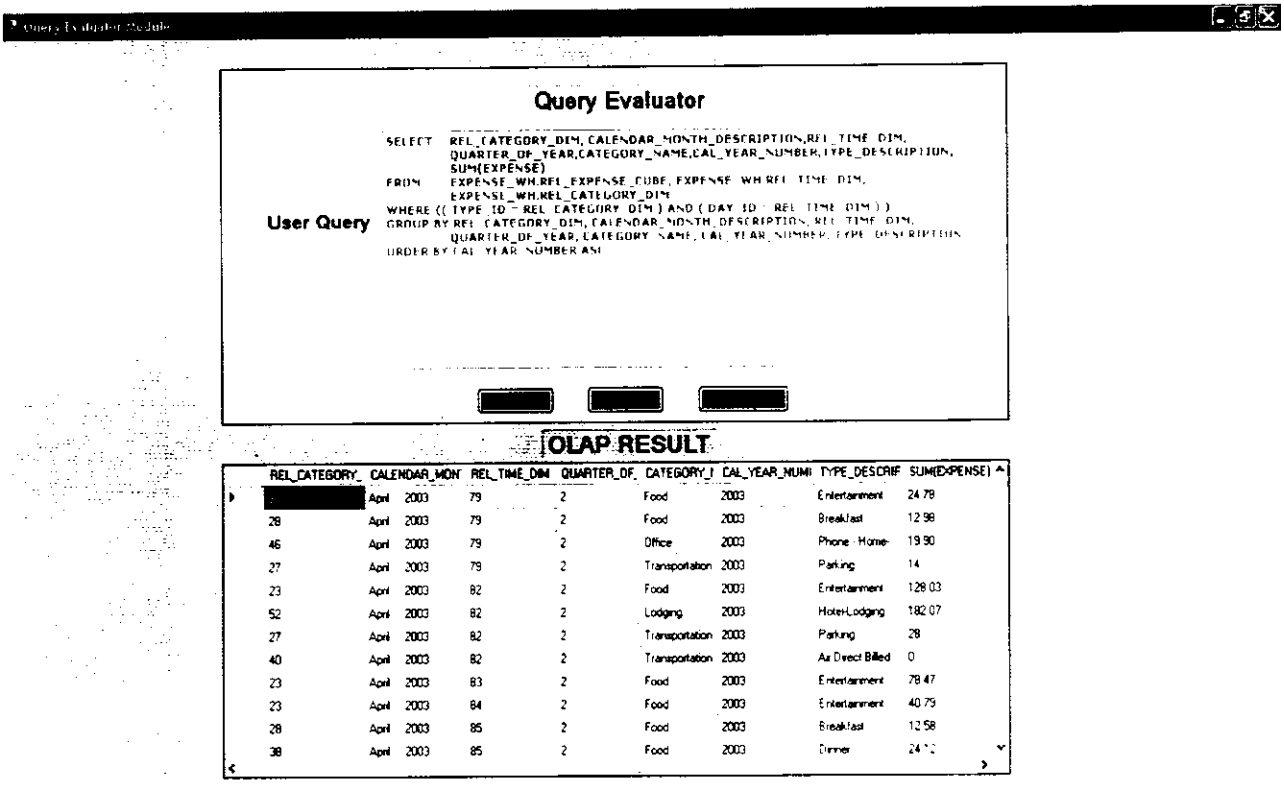


Fig. 6.16. View of Query Evaluator and User Interface Module with OLAP query to OLAPTRAIN

6.4 Discussion and Evaluation:

This section describes the discussion on the results of the experiments performed on different datasets to evaluate the performance of the proposed architecture "MAUDXC".

6.4.1 Result 1:

The set of OLAP queries are executed on the subset of ERRA data for evaluation of the performance of query manager of the proposed architecture "MAUDXC". The following figures represent the execution time and the number of records retrieved by the OLAP query. In Fig. 6.17 the OLAP queries are represented in X-axis and query execution time in seconds is represented in Y-axis. In Fig. 6.18 the OLAP queries are represented in X-axis and total record return by the query is represented in Y-axis.

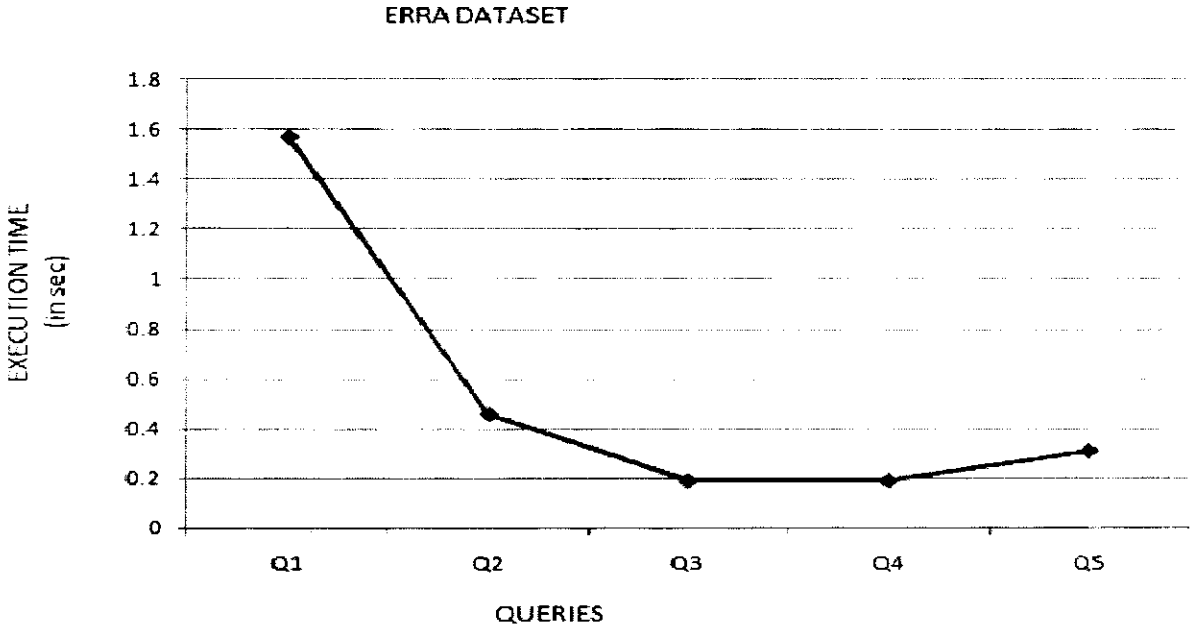


Fig. 6.17 Execution time of queries on ERRA

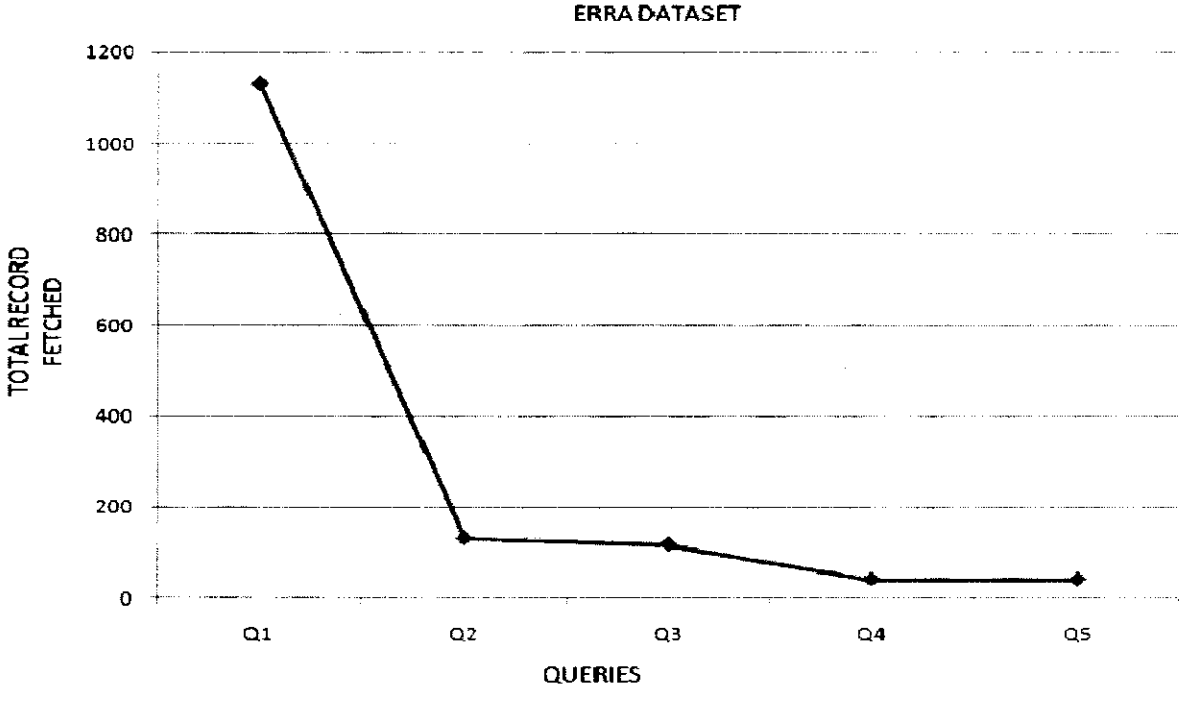


Fig. 6.18 Records retrieve by the queries on ERRA

The above figures clearly represent the high performance of the architecture MAUDXC through its efficient execution and response time of OLAP queries.

6.4.2 Result 2:

The set of OLAP queries are executed on the subset of EXPENSE_WH data for evaluation of the performance of query manager of the proposed architecture "MAUDXC". The following figures represent the execution time and the number of records retrieved by the OLAP query.

In Fig. 6.19 the OLAP queries are represented in X-axis and query execution time in seconds is represented in Y-axis. In Fig. 6.20 the OLAP queries are represented in X-axis and total record return by the query is represented in Y-axis.

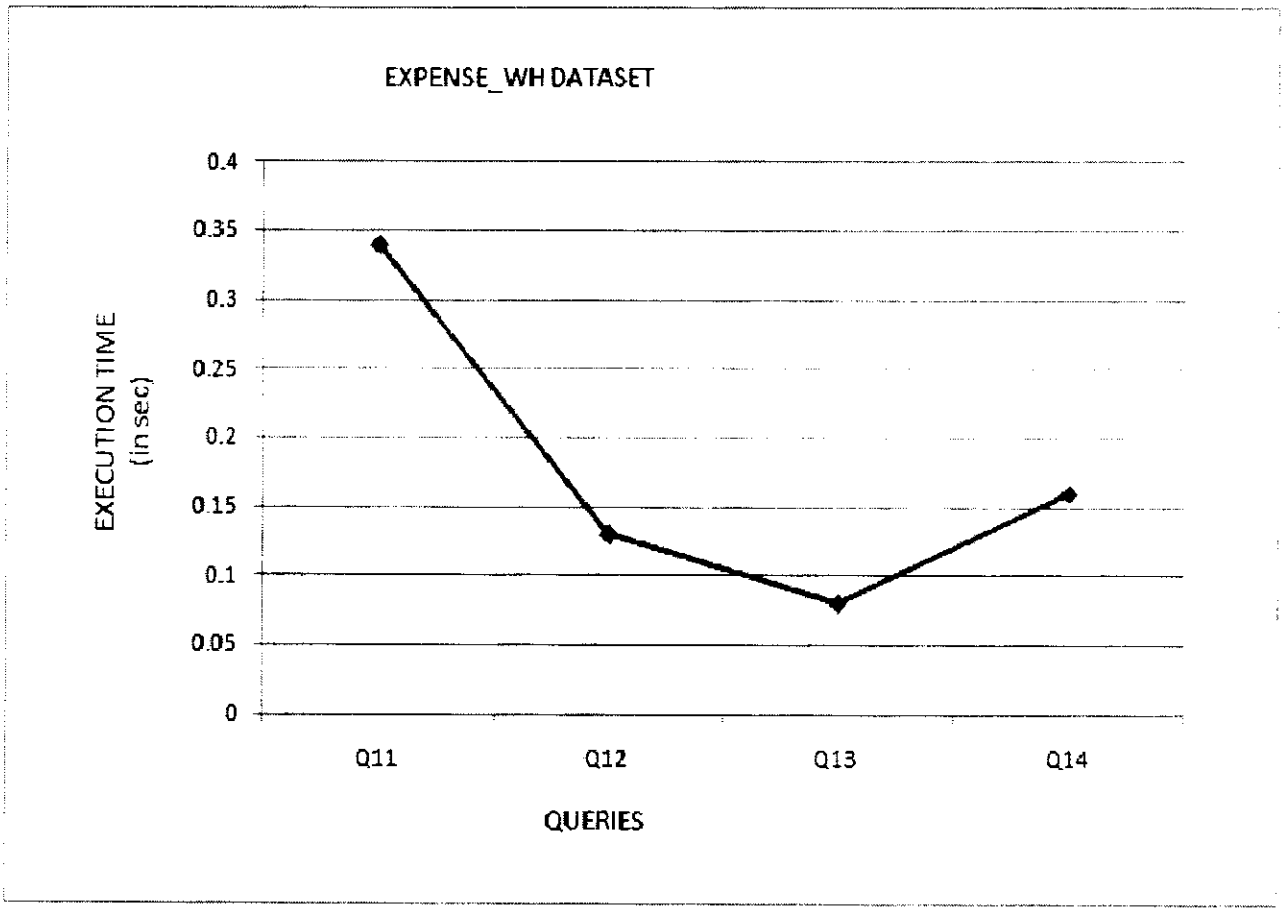


Fig. 6.19 Execution time of queries on Expense_WH

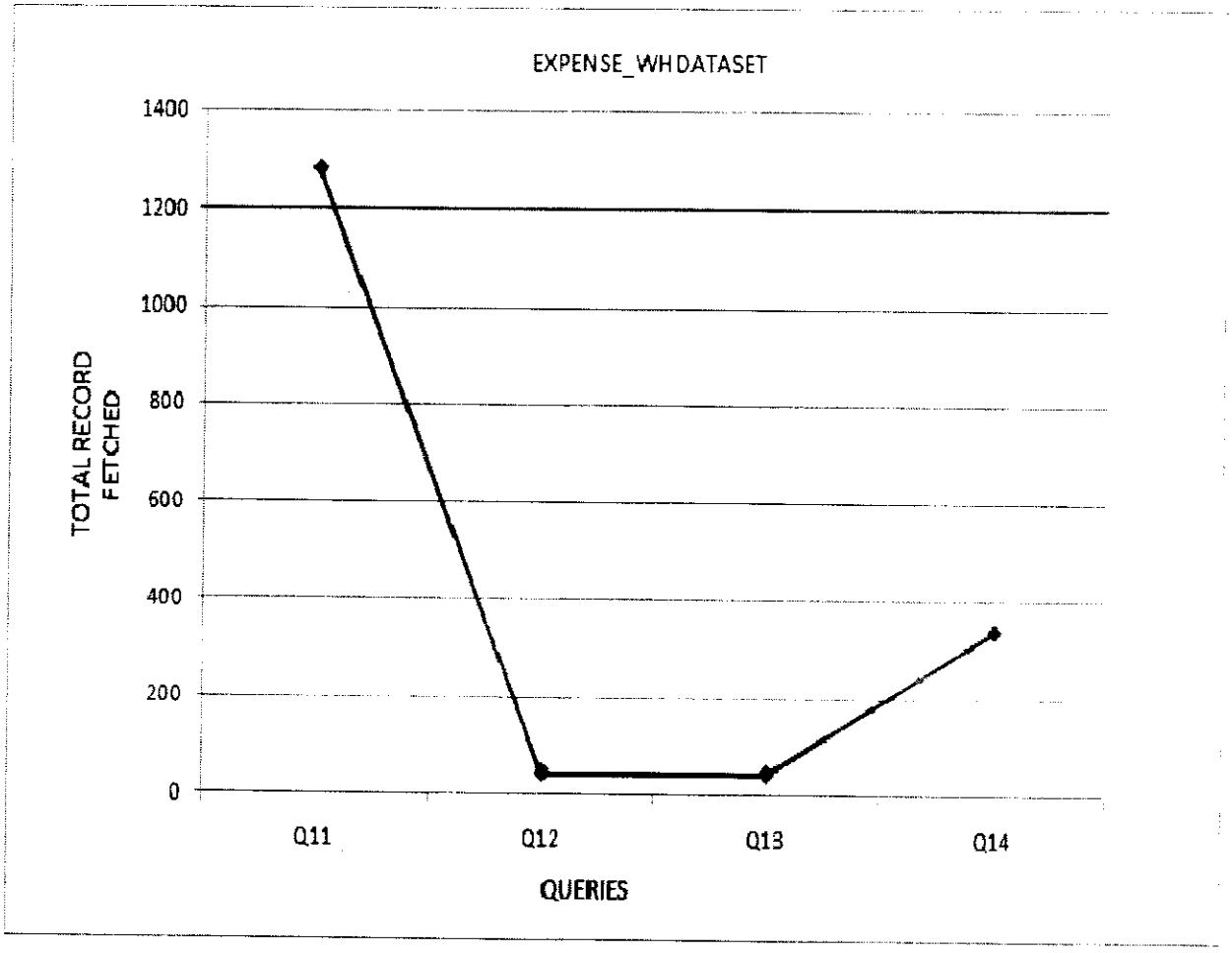


Fig. 6.20 Records retrieve by the queries on Expense_WH

The queries executed on the distributed cube which is stored in location site2. The above figures clearly indicate the efficient evaluation of OLAP queries performed on the distributed data.

6.4.3 Result 3:

The set of OLAP queries are executed on the subset of OLAPTRAIN data for evaluation of the performance of query manager of the proposed architecture "MAUDXC". The following figures represent the execution time and the number of records retrieved by the OLAP query. In Fig. 6.21 the OLAP queries are represented in X-axis and query execution time in seconds is represented in Y-axis. In Fig. 6.22 the OLAP queries are represented in X-axis and total record return by the query is represented in Y-axis.

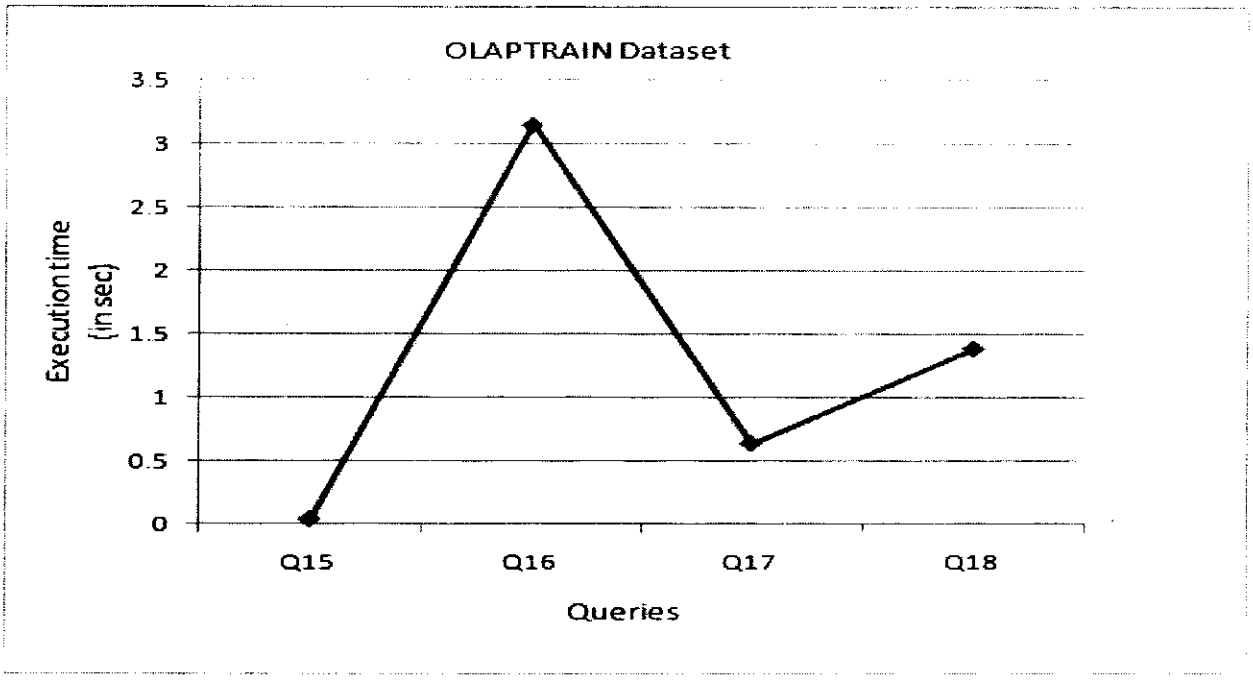


Fig. 6.21 Execution time of queries on OLAPTRAIN

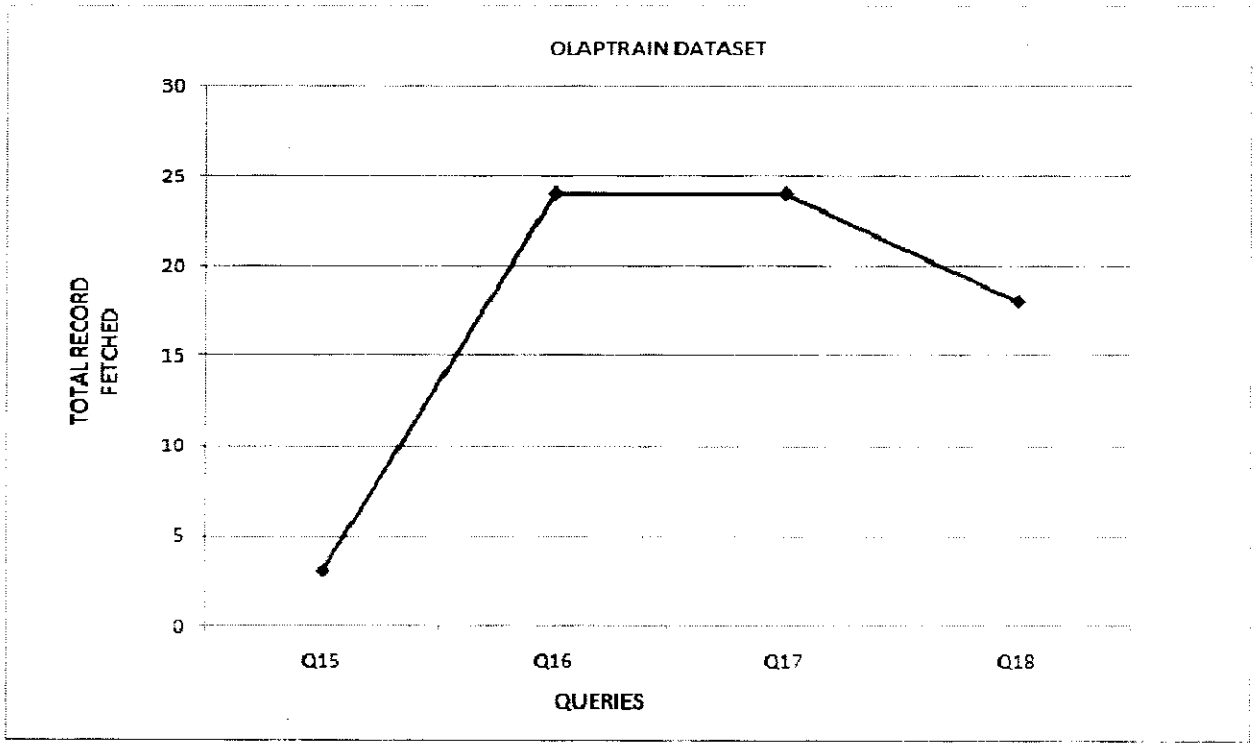


Fig. 6.22 Records retrieve by the queries on OLAPTRAIN

The above figures clearly show the efficient performance of the architecture MAUDXC in term of execution and response time of OLAP queries.

6.4.4 Result 4:

The set of OLAP queries are executed on the subset of ERRA, EXPENSE_WH and OLAPTRAIN data for evaluation of the performance of query manager of the proposed architecture "MAUDXC". The following figures represent the execution time and the number of records retrieved by the OLAP query. In Fig. 6.23 the OLAP queries are represented in X-axis and query execution time in seconds is represented in Y-axis. In Fig. 6.24 the OLAP queries are represented in X-axis and total record return by the query is represented in Y-axis. These queries are performed on the data stored locally and globally as distributed data.

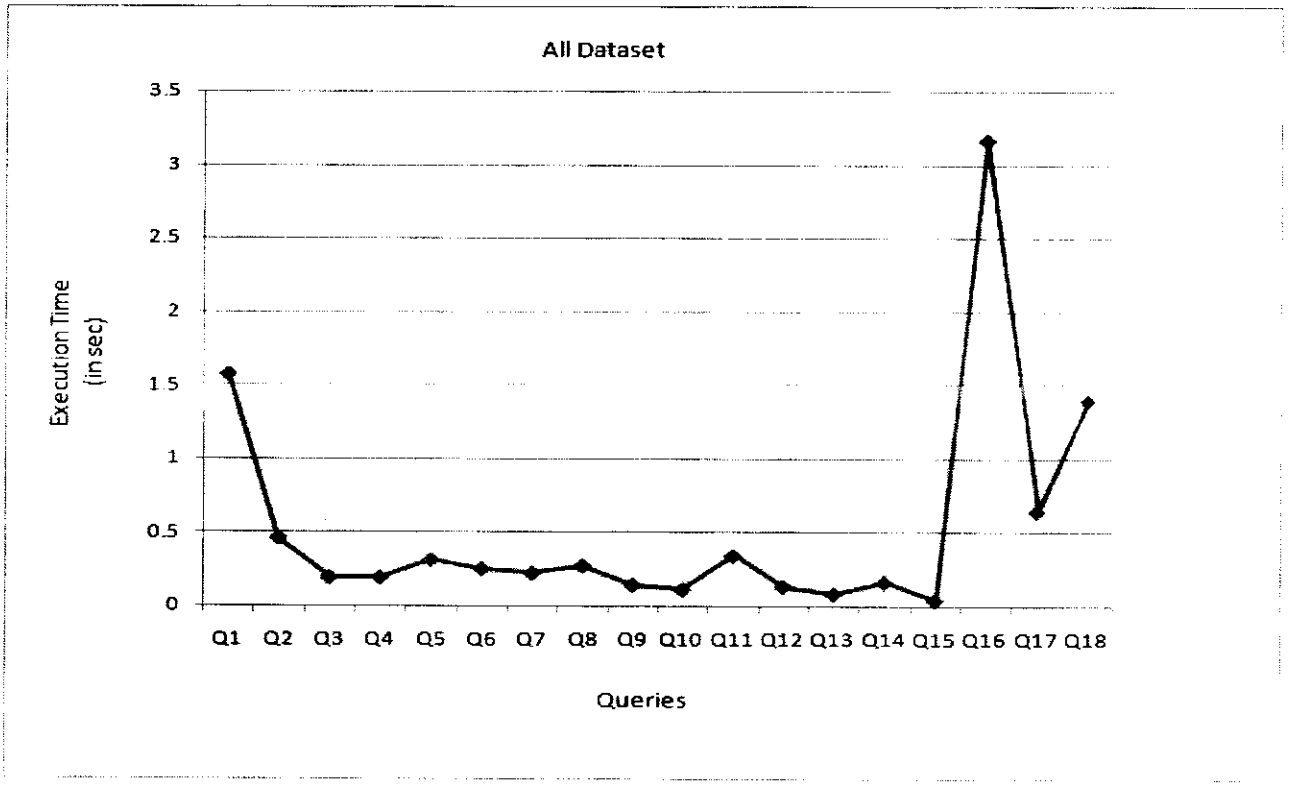


Fig. 6.23 Execution time of all queries on entire dataset

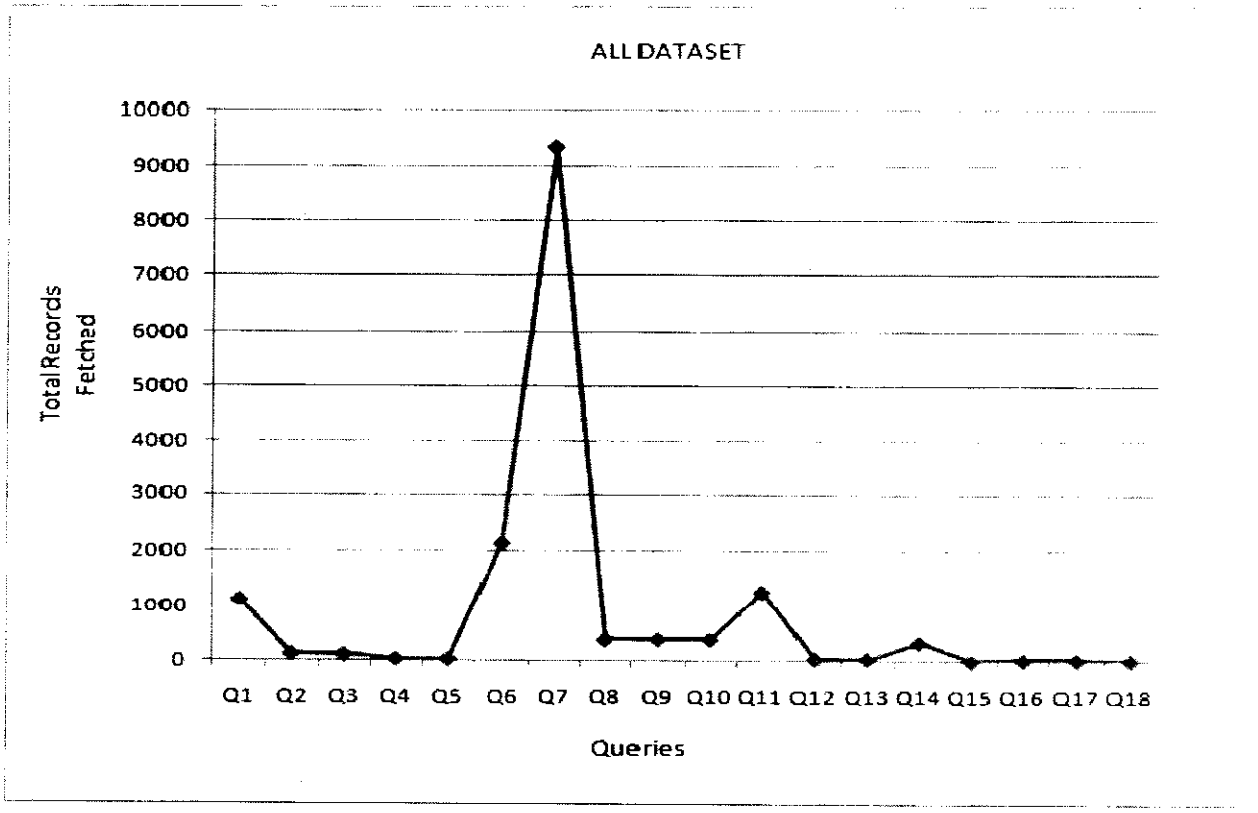


Fig. 6.24 Records retrieve by all queries on entire dataset

The above figures clearly indicate the high performance of the architecture MAUDXC through its efficient execution and response time of OLAP queries.

6.5 Comparison:

This section will provide the comparison between the proposed query evaluation architecture "MAUDXC" and the previous query evaluation architectures called federation manager presented by Pedersen [2].

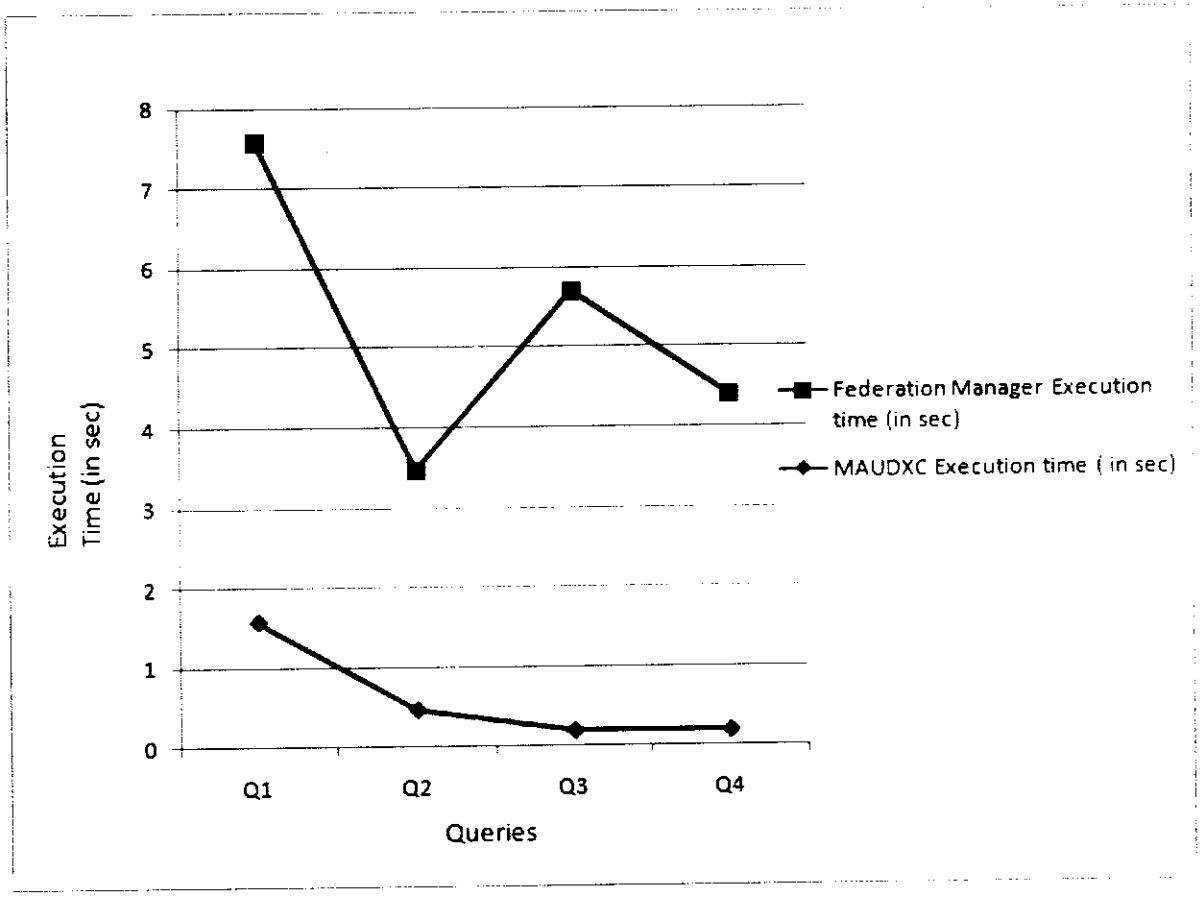


Fig. 6.25 Execution time of queries on MAUDXC and Federation Manager

The set of OLAP queries that are executed on MAUDXC and Federation Manager architecture presented by Pedersen [2]. The Fig. 6.25 represents the execution time of the OLAP queries in seconds. The OLAP queries are represented in X-axis and query execution time in seconds is represented in Y-axis. The above graph depicts that MAUDXC provides more efficient evaluation of the OLAP queries than the Federation Manager.

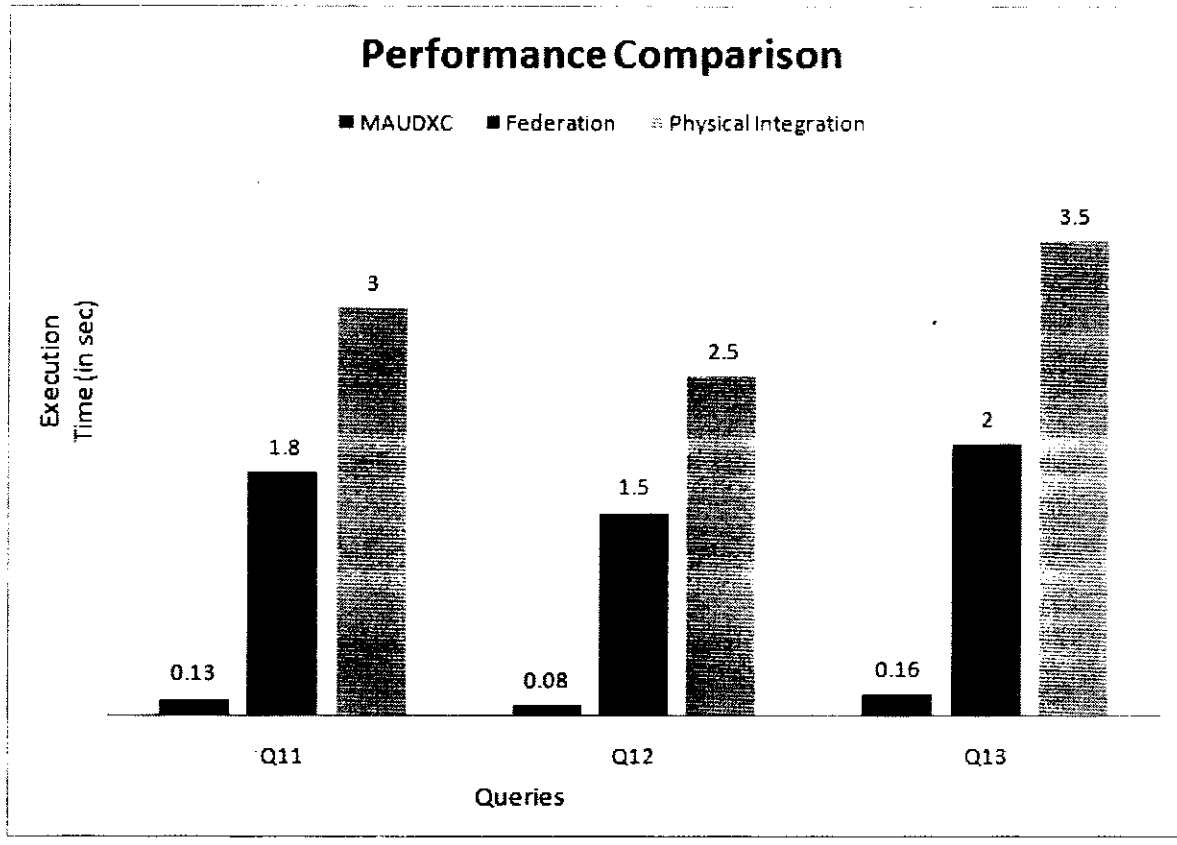


Fig. 6.26 Queries Execution Time Performance MAUDXC, Federation and Physical Integration

The graph presented in Fig. 6.26 indicates the performance comparison of query execution time between our architecture MAUDXC and the published results of Pedersen [2]. This graph clearly depicts the high performance in term of execution time of OLAP queries performed on MAUDXC.

Conclusions and Future Work

7. Conclusions and Future Work

We have proposed a multi-layered architecture of uniformed distributed XOLAP cubes called "MAUDXC". The MAUDXC can effectively deals with the major challenges such as heterogeneity issues, dimension hierarchies, correct aggregation of data and answering of complex OLAP queries that may arise when heterogeneous data sources are integrated to construct OLAP cubes for the handling of OLAP queries. We have extended the existing XOLAP techniques presented by Wang et al. [7], Hachicha et al. [11], Frank et al. [19] and Nippila et al, [21] by adding the support of heterogeneous data sources along with handling of simple and complex OLAP queries all at same architecture. We have also extended the concept of global cube integration by Frank et al, [19] in further two different directions such as global cube integration from uniformed sparse domain cubes and global cube integration from uniformed distributed cubes. The extensions can drastically improve the efficiency of complex OLAP queries. The strength of the MAUDXC architecture is the query manager and its components. The query manager evaluates and executes the OLAP query with very high performance which is indicated through the results of experiments and their comparison with previous query evaluators of Pedersen et al. [2002] described in validation and evaluation section.

This architecture MAUDXC opens further research areas such as

- Extension of query analyzer to analyze OLAP query in any format
- Extension in query evaluator for query re-write, if query analyzer is extended,
- Use of optimization techniques for efficient query processing of OLAP queries when dealing with MOLAP cubes.

The implementation part of the integration of global virtual cube from uniformed distributed cubes and uniformed sparse domain cubes is also left for the future work. The efficient integration of the global cube is another area that can be investigated in future.

References

8. References

[1] T. Niemi, M. Niinimaki, J. Nummenmaa and P. Thanisch," Constructing an OLAP Cube from Distributed XML Data," *Journal of Information Science*. Vol. 31(3) .pp 209-229. November 2002

[2] D. Pedersen, K. Riis and T. Pedersen," "Query optimization for OLAP-XML federations," *in Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP*, New York, USA, pp. 57-64. 2002.

[3] D. Pedersen and T. Pedersen,"Achieving Adaptively For OLAP_XML Federations," ACM New Orleans, Louisiana, USA, November 2003.

[4] X.Yin and T. Pedersen,"Evaluating XML_Extended OLAP Queries Based on Physical Algebra," ACM Washington, DC, USA, November 2004.

[5] R. Bordawekar and C.A. Lang, "Analytical Processing of XML Documents: Opportunities and Challlneges," SIGMOD Record. Vol. 34, No.2. Jun 2005.

[6] B. Park, H. Han and H. Song,"XML_OLAP: A Multidimensional Analysis Framework for XML Warehouses," DaWak, LNCS 3589, pp.32-42. 2005

[7] H. Wang, J. Li, Z. He and H. Gao,"OLAP for XML Data," *in proc. of 5th IEEE IntI conf. on Computer and information technology (CIT'05)* China, IEEE 2005.

[8] F. Jian, J. Pei and A. Fu,"IX_Cube: Iceberg Cubes for Data Warehousing and OLAP on XML Data," *in proc.ACM IntI conf. on Information and Knowledge Management (CIKM)*, ACM Lisboa, Portogal, pp.987-803. 2007

[9] N. Wiwatwattana, H. Jagadish, L. Lakshmanan and D. Srivastava, "X^3: A Cube Operator for XML OLAP," IEEE, 2007

[10] A. Fari, A. Aldin and Y. Helmy, " Designing new XML Based Multidimensional Messaging Interface for the new Xwarehouse Architecture," *in proc. IEEE. IntI multi conf on Computer Science and Information Technology*, pp.525-534, IEEE 2008.

[11] M. Hachicha, H. Mahboubi and J. Darmont,"Expressing OLAP Operators With the TAX XML Algebra," ACM Nantes, France, 2008.

[12] O. Boussaid, R. B. Messaoud, R. Choquet and S. Anthord, "X-Warehousing: An XML Based Approach for Warehousing Complex Data," *International Journal of Web Engineering and Technology*, Vol. 4. No.4, 2008.

[13] W. Hümmer, A. Bauer, and G. Harde., "X- Cube-XML for Data Warehouses. *In Proceedings of the 6th ACM international Workshop on Data Warehousing and OLAP"*, November 2003.

[14] S. Chaudhuri and U. Dayal, "An Overview of data Warehousing and OLAP Technology," ACM Sigmod Record, Vol. 26, No.2, 1997.

[15] I. S. Mumick, D. Quass and B. S. Mumick, "Maintenance of Data Cubes and Summary Tables in a Warehouse", ACM Sigmod Record, Vol. 26, No.2, 1997.

[16] J. Pokarny, "XML Data Warehouse: Modeling and Querying", *in Databases and Information System II*, H. M. Haav, A. Kalija, Ed. Netherland, Kluwer Academic Publishers, pp. 67-80. 2002

[17] B. Vrdolijak, M. Banek and S. Rizzi, "Designing Web Warehouses from XML Schemas", *in Data warehouse and Knowledge Recovery*, Vol. 2737, Y. kambayashi, M. Mohania, W. Wob, Ed. Heidelberg, Springer-Verlag Berlin, pp. 89-98. 2003.

[18] M. Golfarelli, S. Rizzi and B. Vrdolijak, "Data warehouse designing from XML sources", *in proc. the 4th ACM intl. workshop on Data Warehousing and OLAP*, ACM Atlanta, Georgia, USA, pp-40-47. 2001

[19] T. S. C. Frank and C. Chen, "Integrating heterogeneous data warehouse using XML technologies", Journal of Information Sciences, Vol.31, pp. 209-229, 2005.

[20] J. Fong, H. Shiu and D. Cheung, "A Relational-XML Data Warehouse for Data aggregation with SQL and XQuery", in Software: Practices and Experience, Vol. 38, No.11, John Wiley & Sons, Inc. New York, NY, USA, pp. 1183-1213. 2008.

[21] T. Nappila, K. Jarvelin and T. Niemi, "A Tool For Data Cube From Structurally Heterogeneous Xml Documents", Journal of American Society for Information Science and Technology, Vol. 59, No. 3, pp.435-449, 2007.

[22] D. Pedersen, K. Riis and T. B. Pedersen, "XML-Extended OLAP Querying", *in proc. of the 14th Intl. Conf. on Scientific and Statistical Database Management*, IEEE Computer Society Washington, DC, USA, pp. 195-206. 2002.

[23] C. Kit, T. Amagasa and H. Kitagawa, "A Development of an XML-OLAP System", pp. 206-212, 2002

[24] R. M. Bruckner, T. W. Ling, O. Mangisengi, A. M. Tjoa, "A Framework for a Multidimensional OLAP Model using Topic Maps", *in proc of WISE (2)*, pp. 109-118, 2001.

[25] T. Niemi, M. Niinimaki, J. Nummenmaa and P. Tanisch, "Applying Grid Technology to XML Based OLAP cube Construction" *In Proceedings Of The 5th International Workshop on Data Warehousing and OLAP.* ACM, USA, 2003.

[26] M. L. Lee, B. C. Chua, W. Hsu and K. L. Tan, " Efficient Evaluation of Multiple Queries on Streaming XML data" ACM McLean, Virginia, USA, November 2002.

[27] M. R. Jensen, T. H. Moller and T. B. Pedersen, "Specifying OLAP cubes on XML Data" Journal of Intelligent Information System, Vol. 17 , pp.255-280, 2001

[28] R. Bourret, C. Bornhovd and A Buchmann, "A Generic Load/Extract Utilty for Data Transfer Between XML Document and Relational Databases" *2nd Intl. Workshop on Advanced Issue of EC and Web-based Information Systems.* California, 2000

[29] J. Trujillo, S. L. Mora and Y. Song. " Applying UML and XML for designing and interchanging information for data warehouses and OLAP applications," Journal of Databases Management, Vol. 15, No. 1, pp. 41-72, 2004.

[30] F.Ravat, O. Teste, R. Tournier and G. Zurfluh, "Designing and Implementing OLAP Systems from XML Documents", New Trends in Data Warehousing and Data Analysis, Springer US, pp 1-21. 2009.

[31] A. Cuzzocrea, "Cubing Algorithms for XML Data", IEEE FlexDBIST, pp .407- 411. 2009,

[32] M. Ykhlef, "On-Line Analytical Processing Queries for eXtensible Mark-up Language", Information Technology Journal, Vol 8, pp 521-528 (4), 2009.

[33] Y. Zhao, T. Ma, and F. liu, "Research on index Technology for Group-by Aggregation Query in XML Cube", Information Technology Journal, Vol 9(1), pp 116-123. 2010.

[34] A. Cuzzocrea and E. Bertino, "A Secure Multiparty Computation Privacy Preserving OLAP Framework Over Distributed XML Data"SAC, 2010.

[35] V. K. Vaishnavi and W. Kuechler, Design Science Research Methods and Patterns: Innovative Information and Communication Technology, USA, Auerbach Publications, pp. 19-22. 2007.

[36] W. H. Inmon, "Building the Data Warehouse", John Walley, 1992.

Appendix A

Code

// Query Manager and UIMS Module

```
private void button3_Click(object sender, EventArgs e) {

// Extracting the entire OLAP query

 string strq = txtSelect.Text;
 int indxno = strq.IndexOf("FROM", 0)+5;
 int indxnos = strq.IndexOf("WHERE", 0);
 string mystr = strq.Substring(indxno, (indxnos-(indxno+1)));

/*************** Query Analyzer Portion Starts Here****************/

//Module-No: 1

/*To Extract the Cube Names*/

                string mystrcube = searchCubes(mystr);

//Module-No: 2
/*To Extract the Measures' Name from the User Input Query*/

                string[] myMeasuresNames = SearchMeasures(strq);

//Module-No: 3
/*To Extract the Dimensions Name from the User Input Query*/

                string mydims = SearchDimensions(mystr);

/*************** Query Analyzer Portion Ends Here****************/
//Module-No: 1
/*To Extract the Cube Names*/
////////////////////////////////// Cubes ////////////////////////////////////

private string searchCubes(string mystr)
     {
        int indxnocube = mystr.IndexOf(".", 0);
        int indxnoscoubes = mystr.IndexOf("CUBE", 0);
        string mystrcube = mystr.Substring(indxnocube + 1, (indxnoscoubes + 3) - indxnocube);
        return (mystrcube);
     }
/*End of>>>>>>>To Extract the Cubes Name from the User Input Query*/

 //Module-No: 2
/*To Extract the Measures' Name from the User Input Query*/
////////////////////////////////// Measures ////////////////////////////////////
```

```
private string[] SearchMeasures(string strq)
    {
      string[] myMeasuresNames = new string[5];
      int indxofSelect = strq.IndexOf("SELECT", 0) + 7;
      int indxofFrom = strq.IndexOf("FROM", 0);
      string mystrformeasures = strq.Substring(indxofSelect, indxofFrom - (indxofSelect) + 1);

      int indxofstartbract = mystrformeasures.IndexOf("(", 0);
      int indxofendbract = mystrformeasures.IndexOf(")", 0);
      myMeasuresNames[0] = mystrformeasures.Substring(indxofstartbract + 1, indxofendbract -
       (indxofstartbract) - 1);
      return (myMeasuresNames);
    }
```

/*End of>>>>>>>To Extract the Measures Name from the User Input Query*/

//Module-No: 3

/*Start of:>>>>>>>>>>>>>>>>>To Extract the Dimensions Name from the User Input Query*/

//////////////////////////////// **Dimensions** ////////////////////////////////////

```
private string SearchDimensions(string mystr)
    {
      string[] myDimNames = new string[5];
      int indxdim = mystr.IndexOf(",", 0);
      //int indxnodim = mystr.IndexOf("DIM", 0);
      int j;
      j = 0;


      for (int i = 0; i < mystr.Length; )
      {
        myDimNames[j] = (myDimNames + mystr.Substring(i, indxdim - i)).ToString();

        if (indxdim == mystr.Length)
          break;
        i = indxdim + 1;
        indxdim = mystr.IndexOf(",", i);
        if (indxdim == -1)
          indxdim = mystr.Length;
        j++;
      }
      string mydims = "";
      for (int i = 0; i <= j; i++)
      {
        if (myDimNames[i].Contains("DIM"))
          mydims = mydims + myDimNames[i];
        if (i != j)
          mydims = mydims + ", ";
      }
      return (mydims);
    }
```

```
/*End of>>>>>>>To Extract the Dimensions Name from the User Input Query*/
/*************** Query Matcher Portion Starts Here****************/

/******** Database Connection ***********/
        String UID = "";
        String PWD = "";
        String dbName = "";
        OracleConnection con = new OracleConnection();
                string constr = "User Id=" + UID + ";Password=" + PWD + ";Data Source=" +
         dbName + ";Persist Security Info=True;" +
              "Min Pool Size=10;Connection Lifetime=120;";
        con.ConnectionString = constr;
        con.Open();

/******** Database Credentials ***********/
        String str = "'";
        string str1 = "%'";
        string localSQL = "";

   try
        {
           localSQL = "SELECT * FROM META_CUBE WHERE CUBE_NAME like " + str +
           mystrcube + str1 AND MEASURES like '" + str + mymeasureNames + str1 AND
           DIMENSIONS like '" + str + mydims+ str1;
          OracleCommand cmdOracle = new OracleCommand(localSQL, con);
          cmdOracle.CommandType = CommandType.Text;
          OracleDataReader drOracle = cmdOracle.ExecuteReader();
          if (drOracle.HasRows)
          {
             MessageBox.Show("Requested Data is Found, Press Execute Button to display Data");
          }
          DataSet ds = new DataSet();
          OracleDataAdapter oda = new OracleDataAdapter(localSQL, con);
          oda.Fill(ds, "cube");
          databName =ds.Tables[0].Rows[0]["db_name"].ToString();

//Extracting details from meta-cube

          uid=ds.Tables[0].Rows[0]["schema"].ToString();
          CUBName = ds.Tables[0].Rows[0]["cube_Name"].ToString();
           meausreName= ds.Tables[0].Rows[0]["measures"].ToString();
          dimName = ds.Tables[0].Rows[0]["dimensions"].ToString();
           schemaName= ds.Tables[0].Rows[0]["schema"].ToString();
          dbName = ds.Tables[0].Rows[0]["database"].ToString();
          locName = ds.Tables[0].Rows[0]["location"].ToString();

        }
        catch
        {
```

```
            throw;
        }

    }
```

/*********** **Query Matcher Portion Ends Here**********/

// Query Executor Portion Starts here

```
private void frmQueryEvaluator_Load(object sender, EventArgs e)
    {

    }

private void btnExecute_Click(object sender, EventArgs e)
    {
      OracleConnection con = new OracleConnection();
       string constr = "User Id=" + uid + ";Password=" + password + ";Data Source=" + databName
       + ";Persist Security Info=True;" +
            "Min Pool Size=10;Connection Lifetime=120;";
      con.ConnectionString = constr;
      con.Open();

      DataSet ds = new DataSet();
      string strq = txtSelect.Text;
      OracleDataAdapter oda = new OracleDataAdapter(strq, con);

      // string strq = txtSelect.Text;

      //oda.Fill(ds, "FINANCIAL_CUBE");
       oda.Fill(ds, uid+"."+CUBName);
       dataGridView1.DataSource = ds.Tables[0];
    }
private void button1_Click(object sender, EventArgs e)
    {
      txtSelect.Text = "";
      dataGridView1.DataSource = null;
    }
```

/*********** **Query Executor Portion Ends Here**********/

//XML Converter Module

```
namespace XmlConverter
{
  public partial class Form1 : Form
  {
    string UID, PWD, dbName;
    public Form1()
    {
      InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {


        OracleConnection con = new OracleConnection();
        string constr = "User Id=" + UID + ";Password=" + PWD + ";Data Source=orcl;Persist
                    Security Info=True;" +
         "Min Pool Size=10;Connection Lifetime=120;";

        con.ConnectionString = constr;
        con.Open();

       using (DataSet myDataSet = new DataSet())
       {



     //Open the Oracle Database Connection
      openConnection();

// add your own query to select the tabel
      OracleDataAdapter dataAdapter = new OracleDataAdapter("Select * from  " +
            cboTableName.SelectedItem.ToString()  , con);
        dataAdapter.Fill(myDataSet);

     using (DataTable myDataTable = myDataSet.Tables[0])
     {
     foreach (DataColumn column in myDataTable.Columns)
     {
     column.ColumnMapping = MappingType.Attribute;
     }

//write your own file name and specify the path
  myDataSet.WriteXml(@"XMLCUBES\"+cboTableName.SelectedItem.ToString()+".xml");

foreach (DataColumn column in myDataTable.Columns)
 {
```

```
        column.ColumnMapping = MappingType.Element;
    }
        myDataSet.WriteXml(@"XML CUBES\"+ cboTableName.SelectedItem.ToString()+".xml");
        lblMessage.Text = "File is saved at Location With Name: C:\\XML
        CUBES\\"+cboTableName.SelectedItem.ToString()+".xml";
  lblMessage.Visible = true;
    }
}
}
```

// Database Connection

```
    private void Form1_Load(object sender, EventArgs e)
    {


    }
    private void openConnection()
    {
      string UID, PWD, dbName;
      UID = txtUID.Text;
      PWD = txtPWD.Text;
      dbName = txtDBName.Text;


            string constr = "User Id=" + UID + ";Password=" + PWD + ";Data
            Source=orcl;Persist Security Info=True;" + "Min Pool Size=10;Connection
            Lifetime=120;";
      try
      {
        OracleConnection con = new OracleConnection();
        con.ConnectionString = constr;
        con.Open();
      }
      catch (Exception)
      {

        throw;
      }
    }

private void button2_Click(object sender, EventArgs e)
    {
      if (txtUID.Text.Length > 0)
      {
        UID = txtUID.Text;
        PWD = txtPWD.Text;
        dbName = txtDBName.Text;
        String str = "'%";
        //string str1 = "EXPENSE_WH.";
        OracleConnection con = new OracleConnection();
```

```
                    string constr = "User Id=" + UID + ";Password=" + PWD + ";Data
                Source="+dbName+";Persist Security Info=True;" + "Min Pool
                Size=10;Connection Lifetime=120;";
        con.ConnectionString = constr;
        con.Open();
        cboTableName.Items.Clear();
        string localSQL = "";
        try
        {
                localSQL = "SELECT TABLE_NAME FROM USER_TABLES WHERE
                TABLE_NAME  LIKE " + str + "_CUBE%'";

    OracleCommand cmdOracle = new OracleCommand(localSQL, con;
    cmdOracle.CommandType = CommandType.Text;

    OracleDataReader drOracle = cmdOracle.ExecuteReader();
     while (drOracle.Read())
     {
                        cboTableName.Items.Add(drOracle.GetOracleValue(0).ToString());
       }
     }
     catch
     {
       throw;
     }
   }
}
```

Appendix B:
Screen Shot

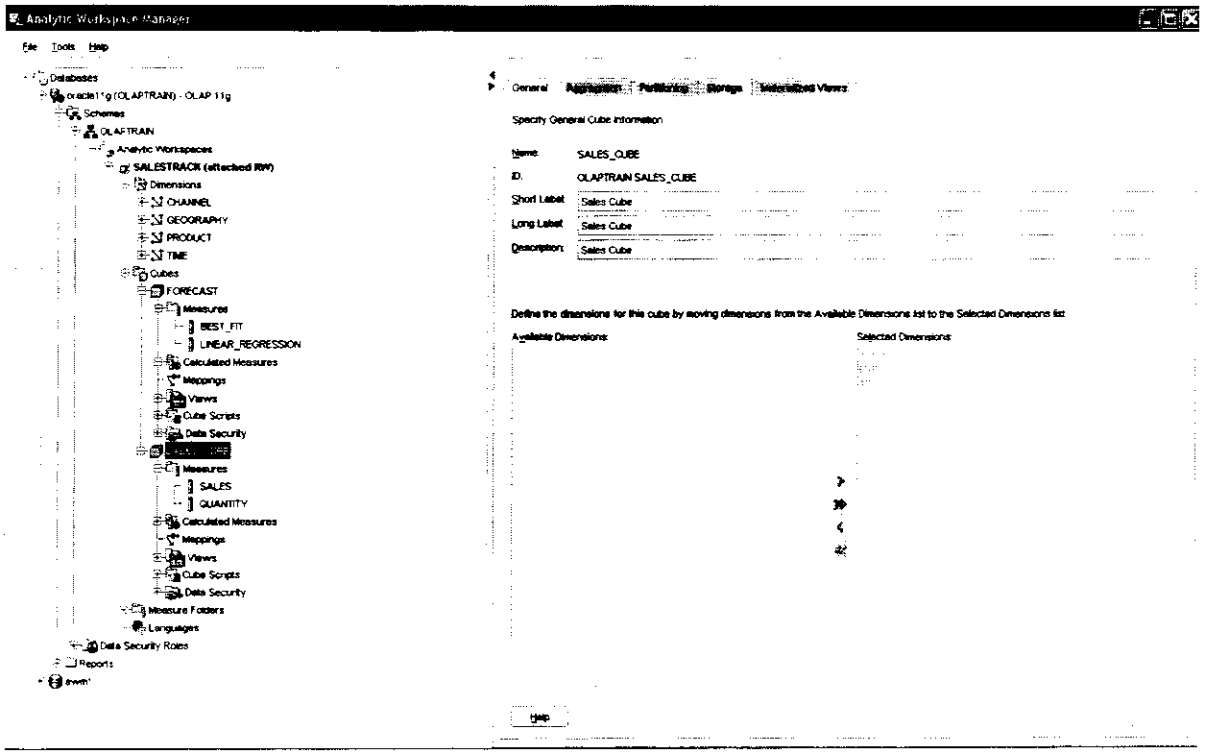


Fig. A View of AWM for ROLAP cubes and dimension by using OLAPTRAIN data set

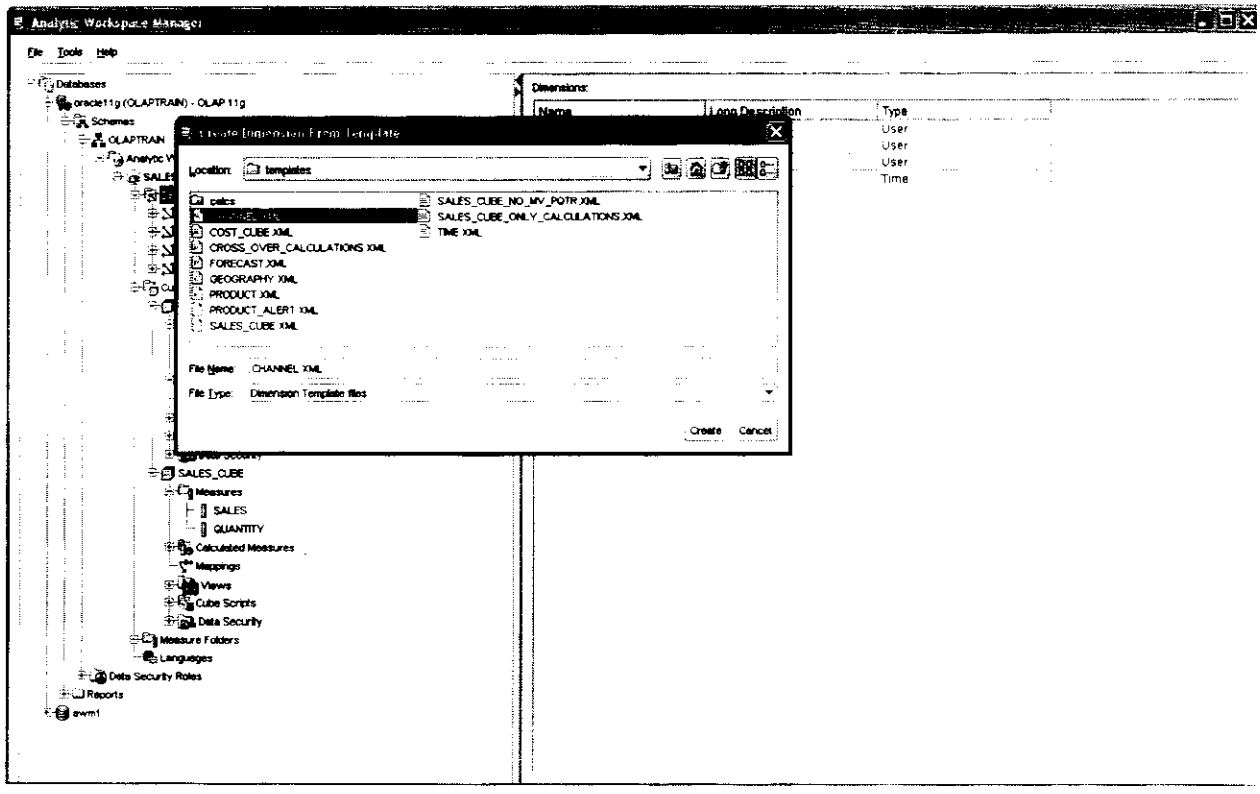


Fig. B View of AWM for defining dimension from XML files

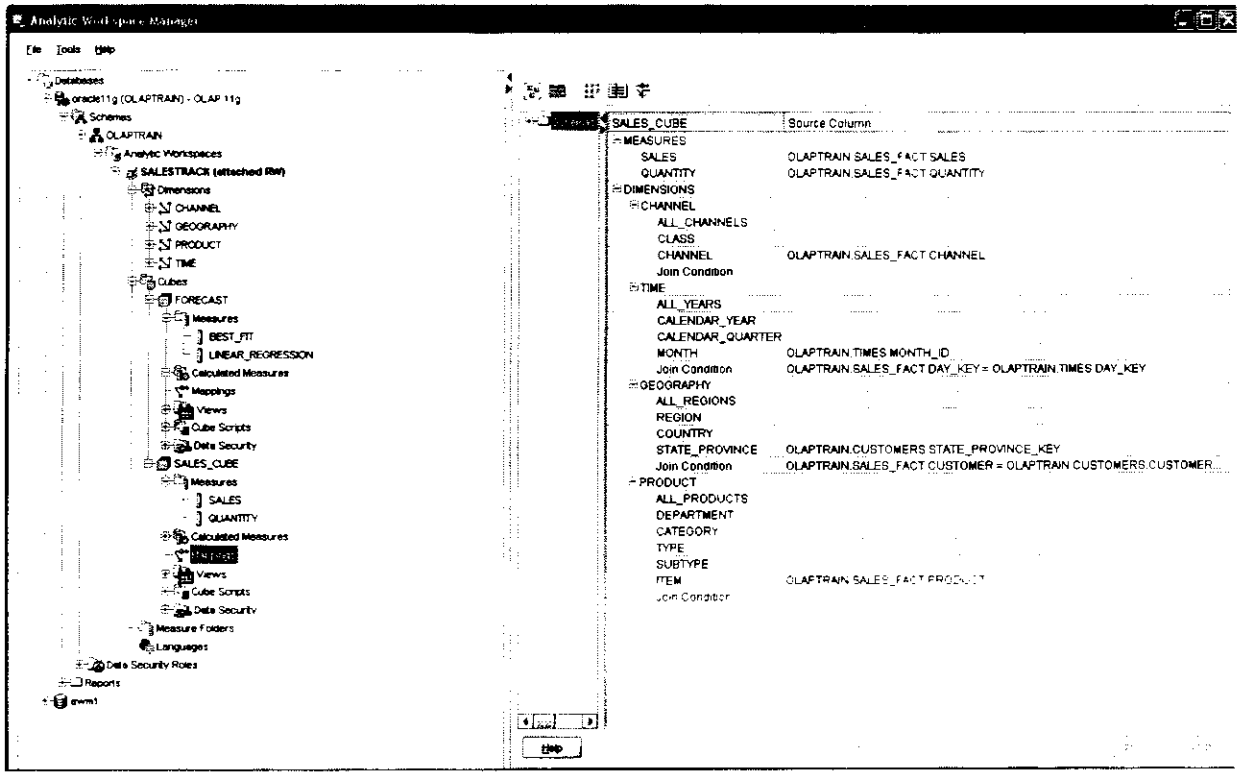


Fig. C View of AWM with mapping of the ROLAP cube "sales_cube"

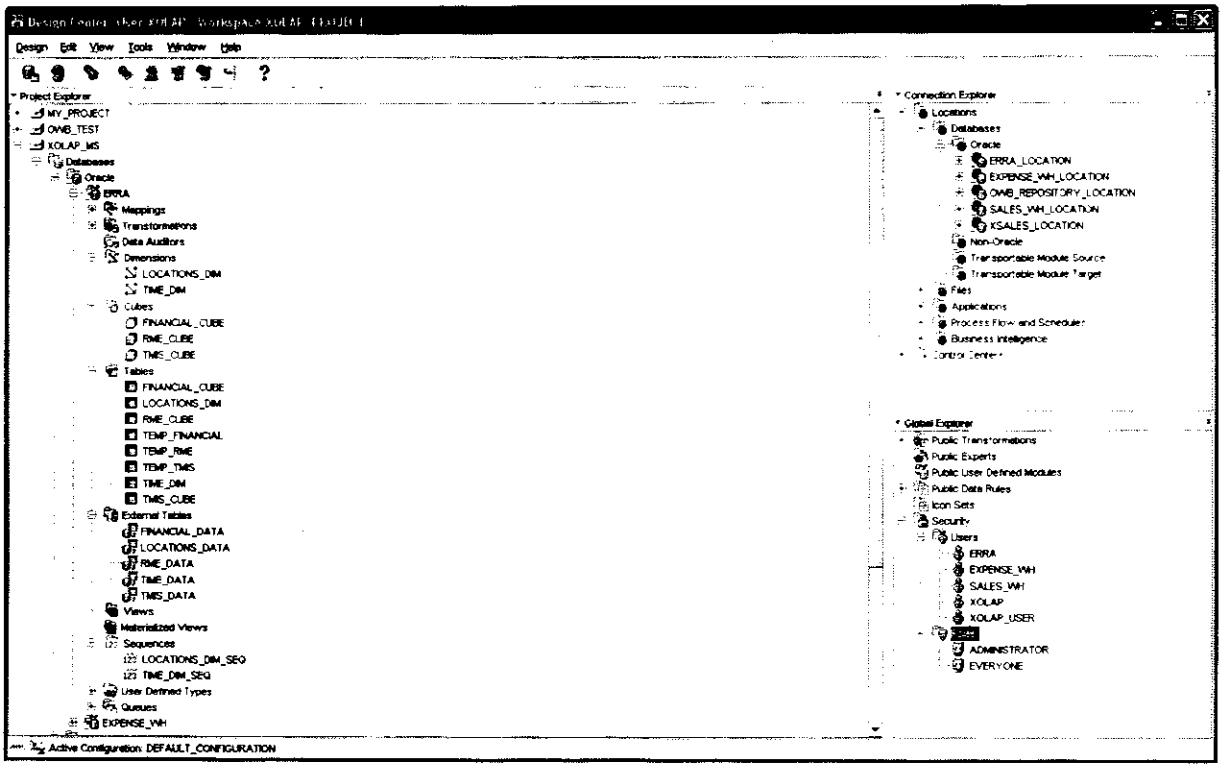


Fig. D View of OWB for ROLAP cubes and dimension by using ERRA data set

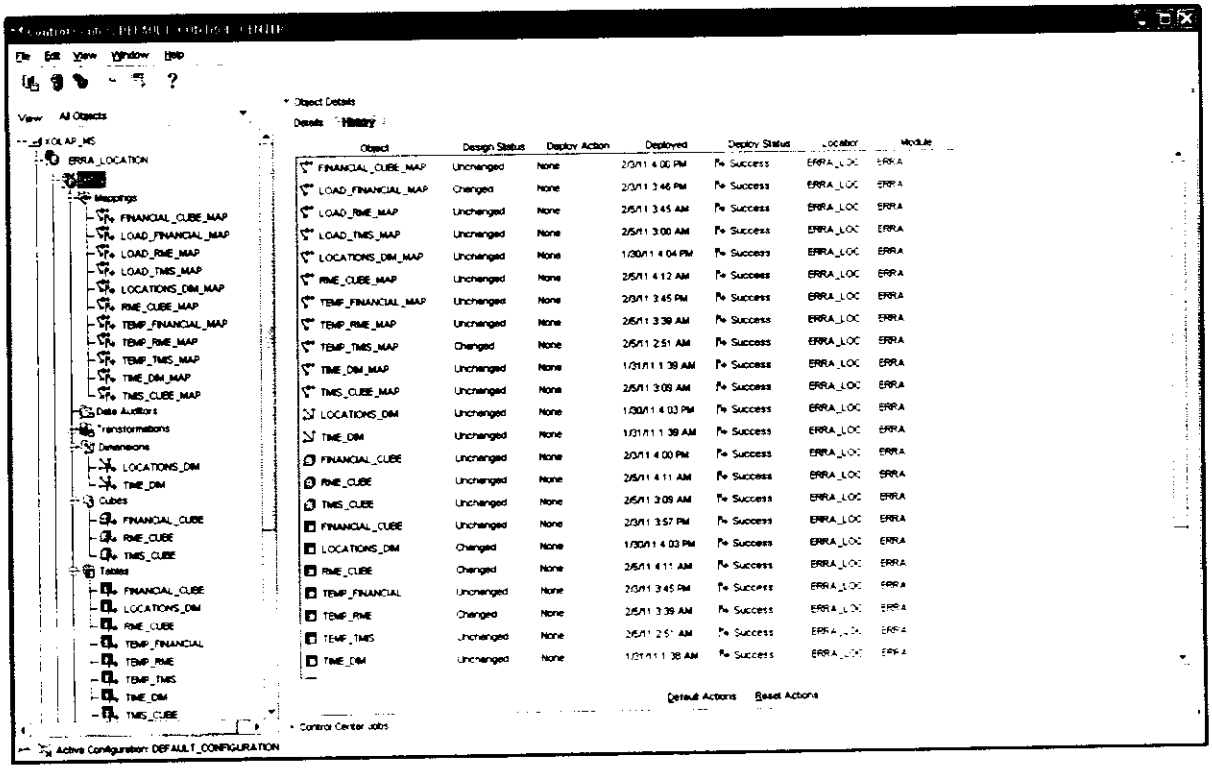


Fig. E View of Control Center of OWB after deploying the mappings, dimensions, cubes, tables, external tables and sequences of ERRA dataset.

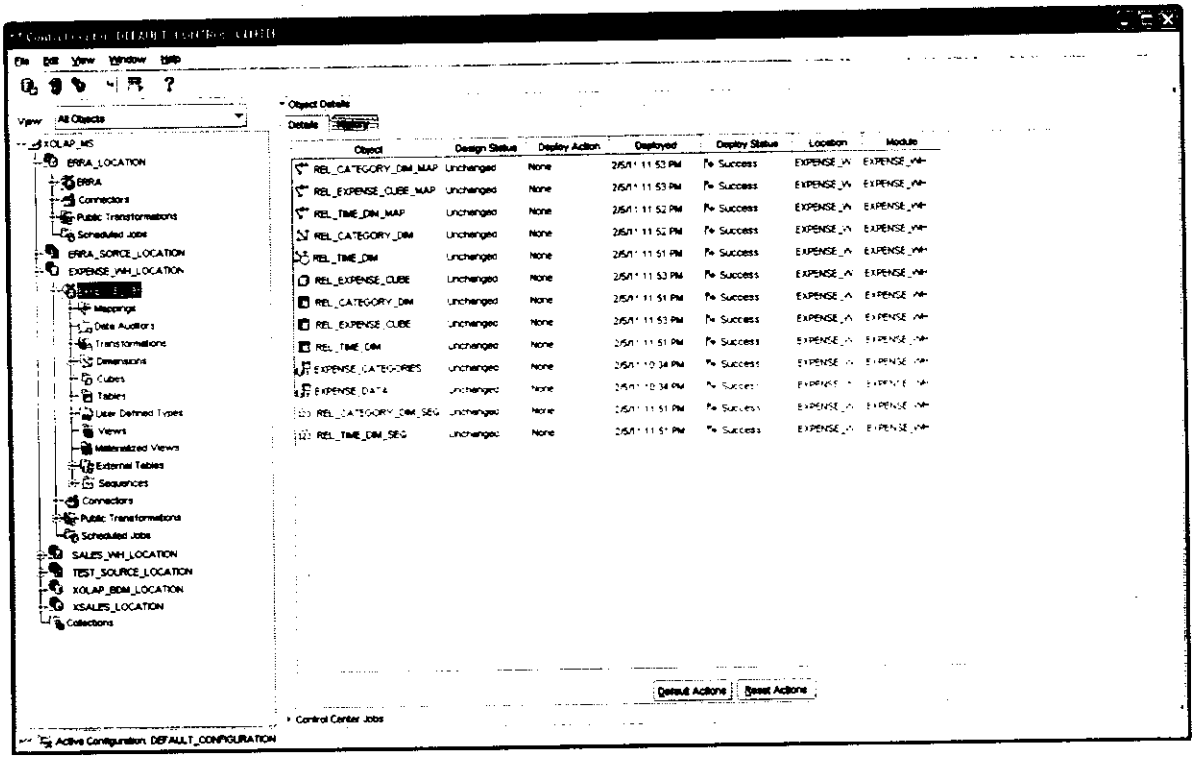


Fig. F View of Control Center of OWB after deploying the mappings, dimensions, cubes, tables and sequences of EXPENSE_WH dataset.

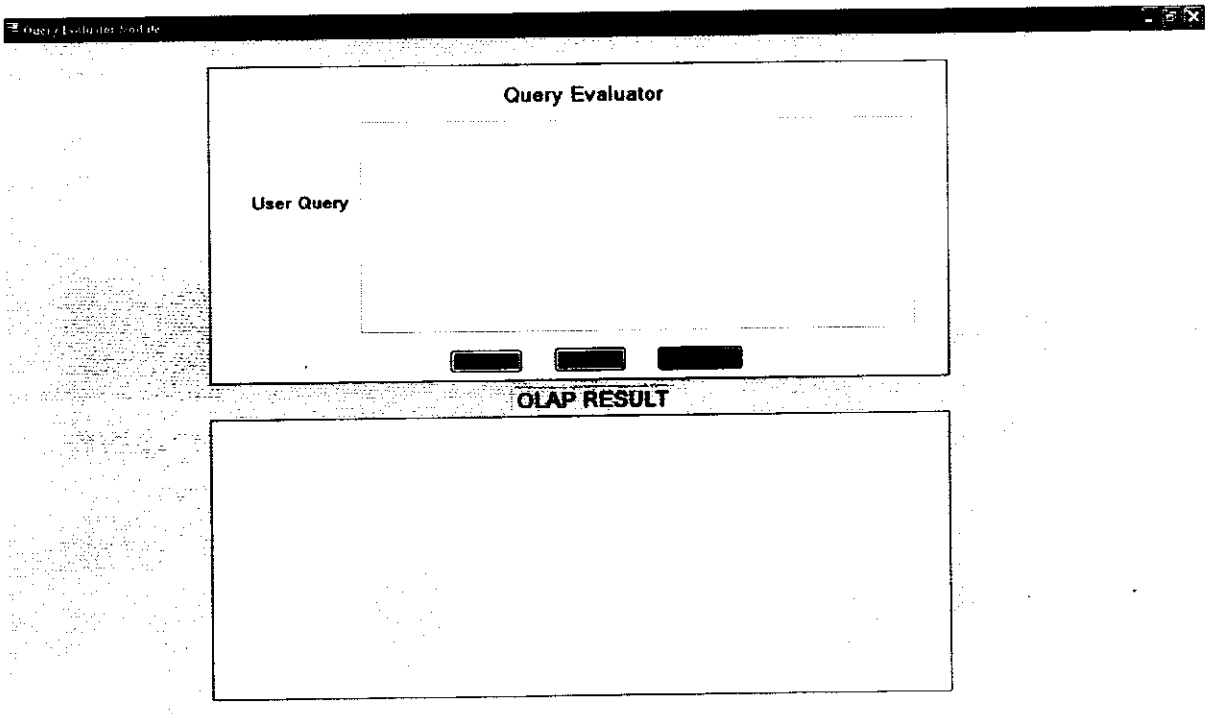


Fig. G View of the Query Evaluator Module

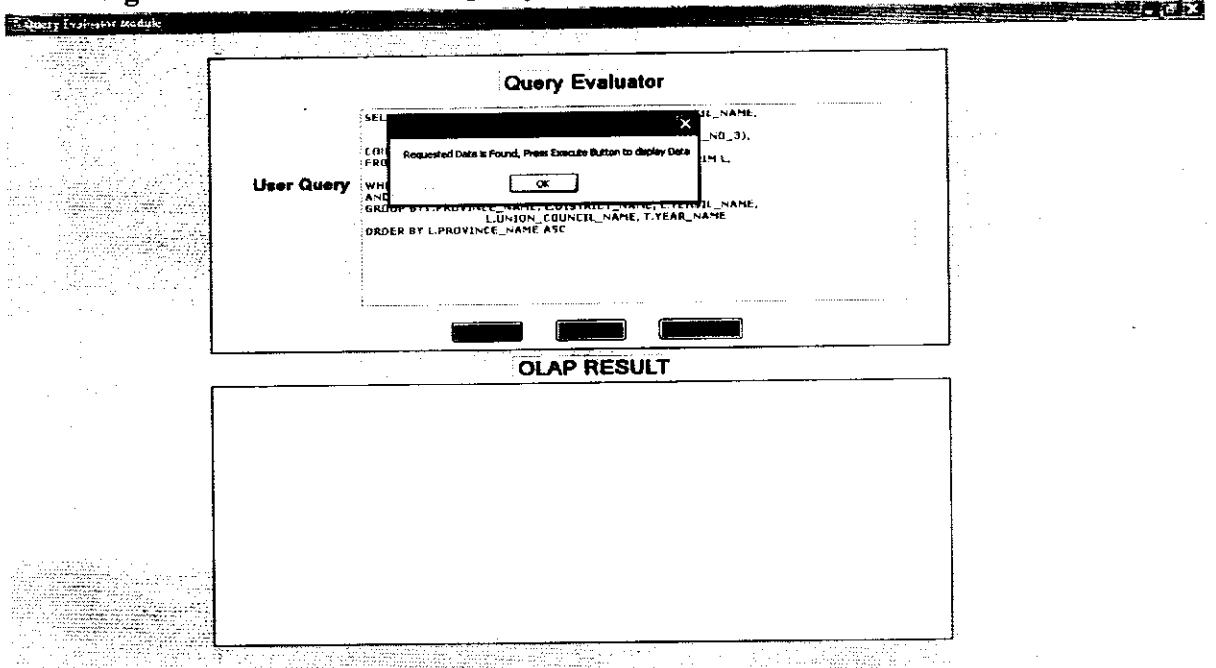


Fig. H View of the Query Evaluator Module while executing OLAP query

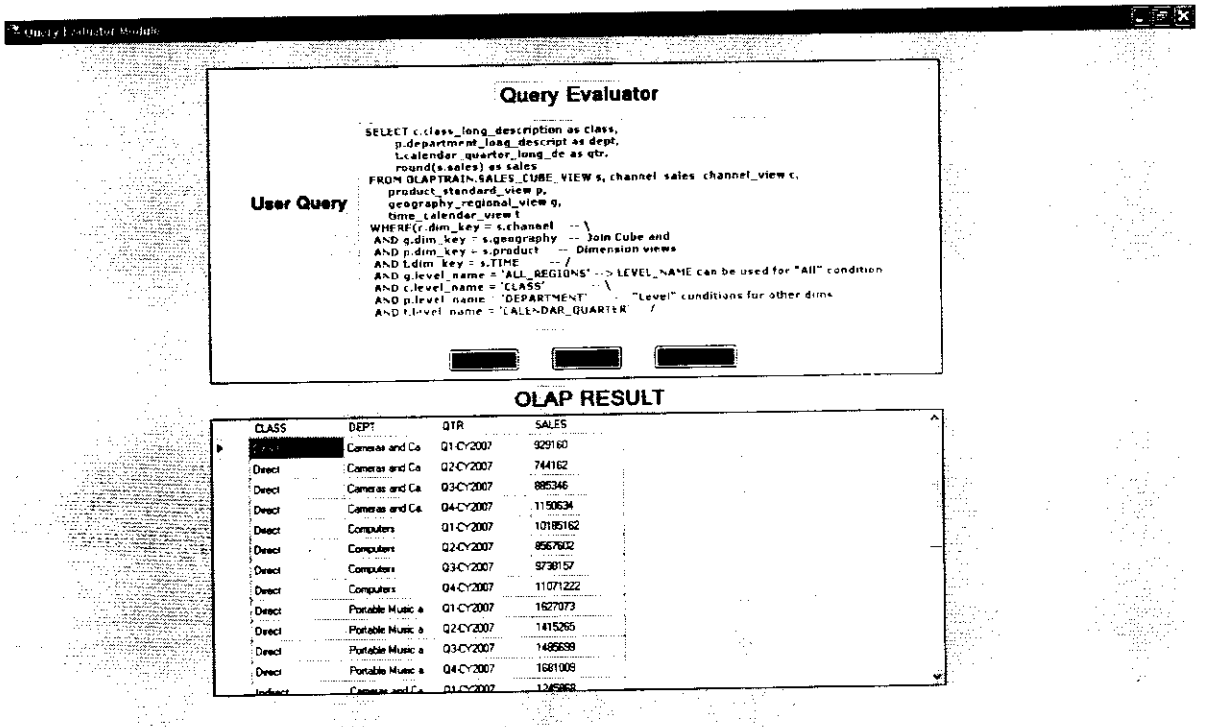


Fig. I View of the Query Evaluator Module after executing the OLAP query with result on OLAPTRAIN Dataset

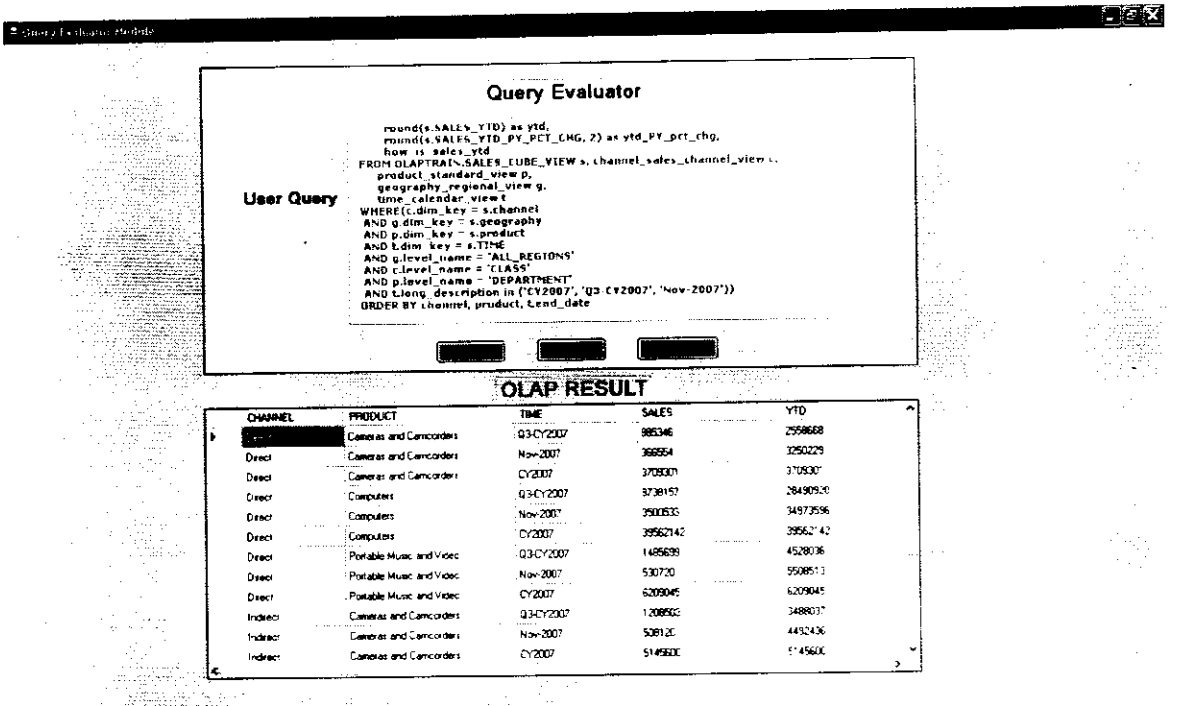


Fig. J View of the Query Evaluator Module after executing the OLAP query with result on OLAPTRAIN Dataset

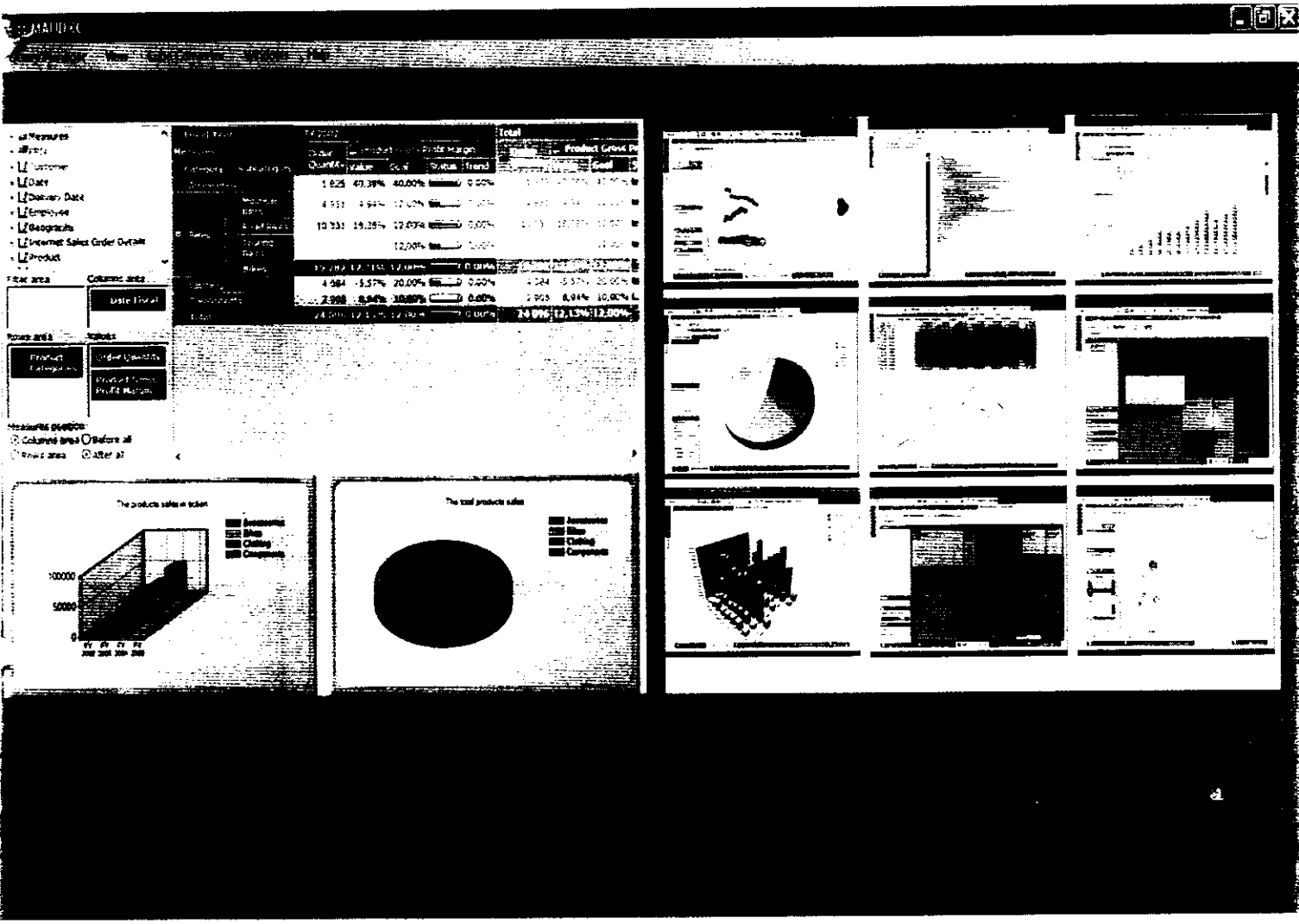


Fig. K View of the Main Screen Window of MAUDXC

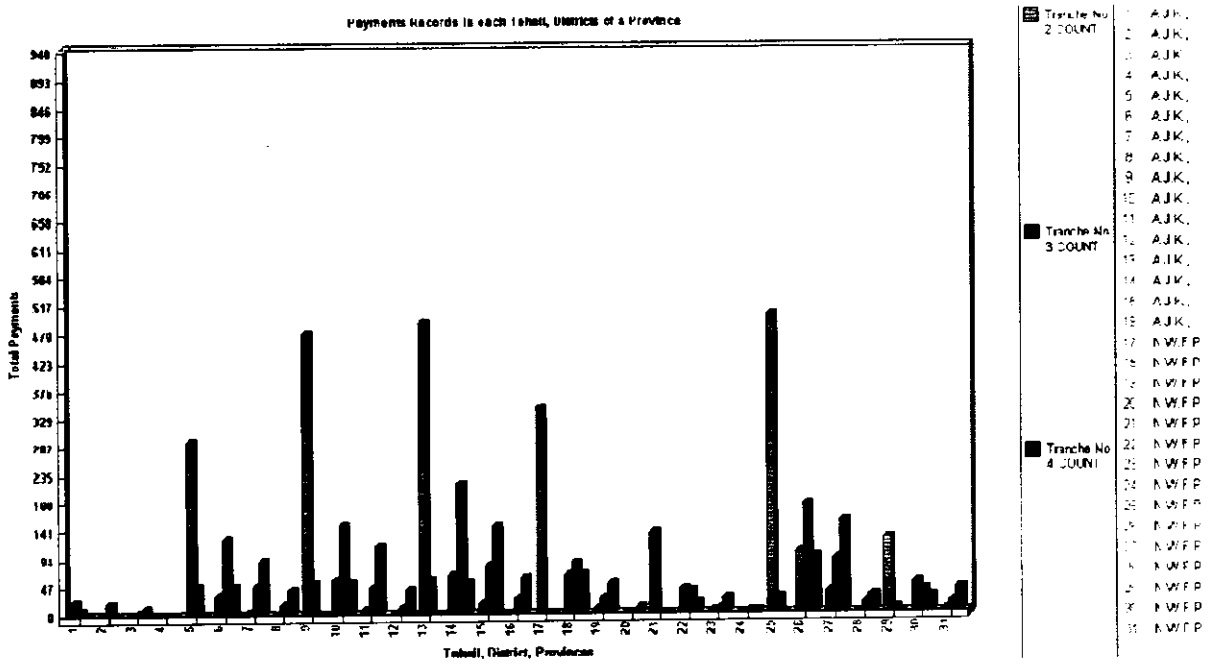


Fig. L Graphical Representation of query result on ERRA-Experiment 1

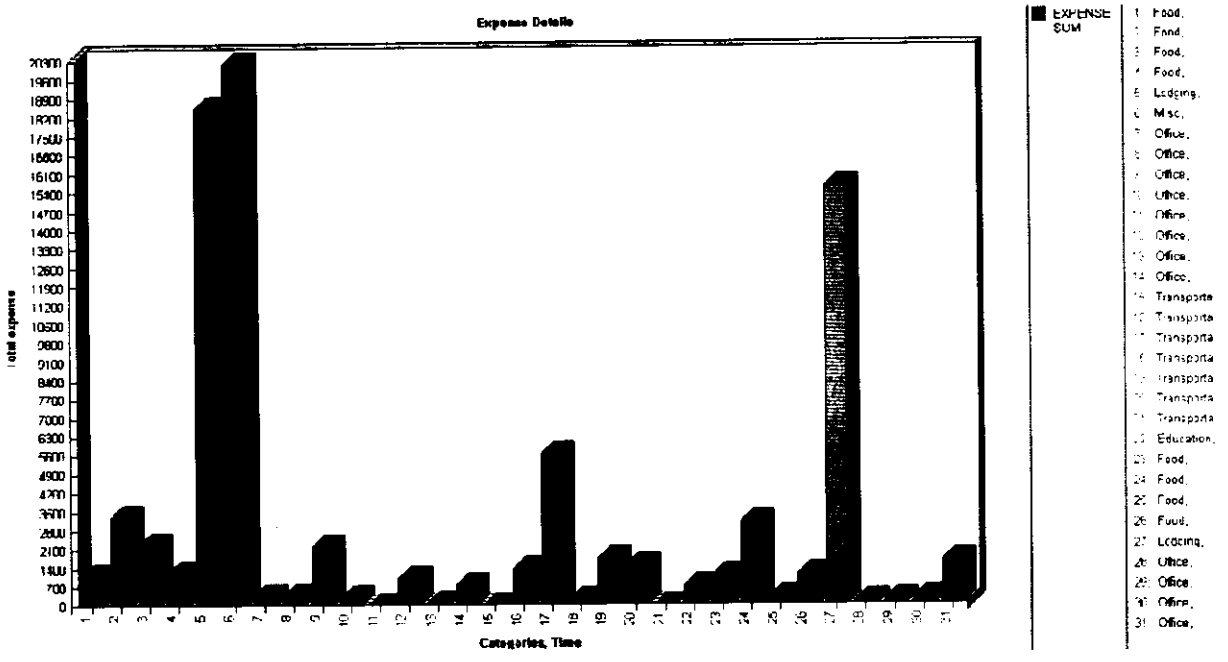
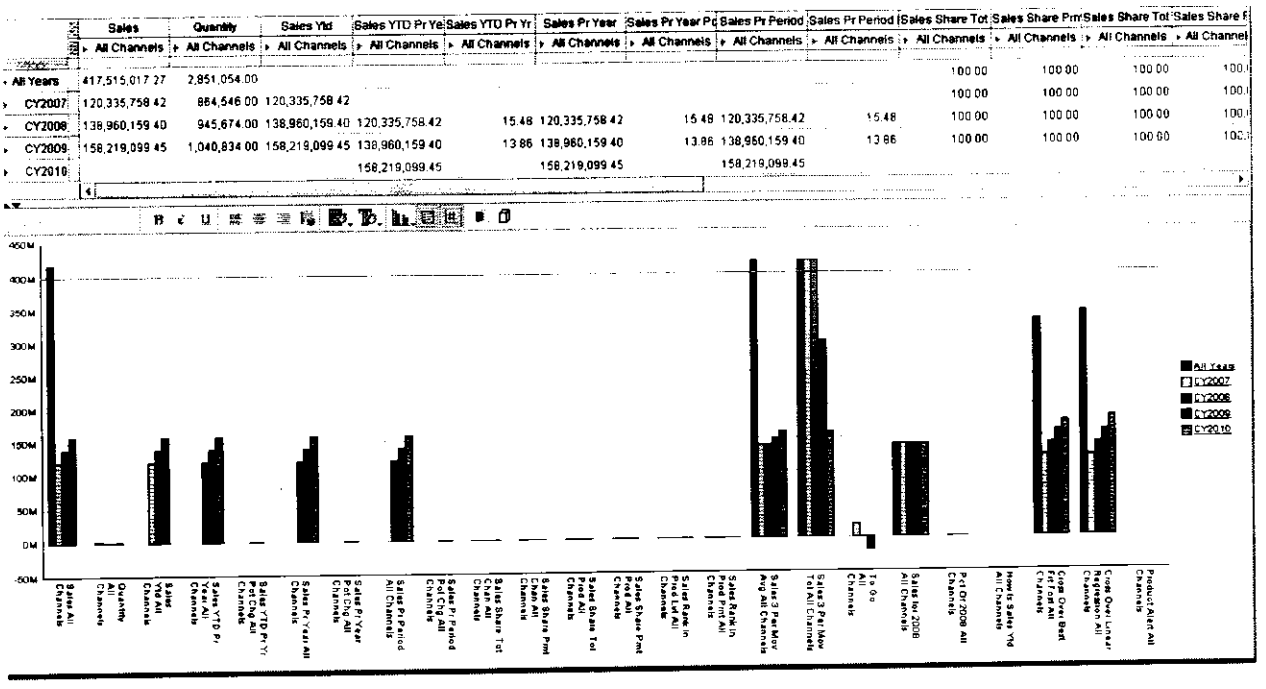
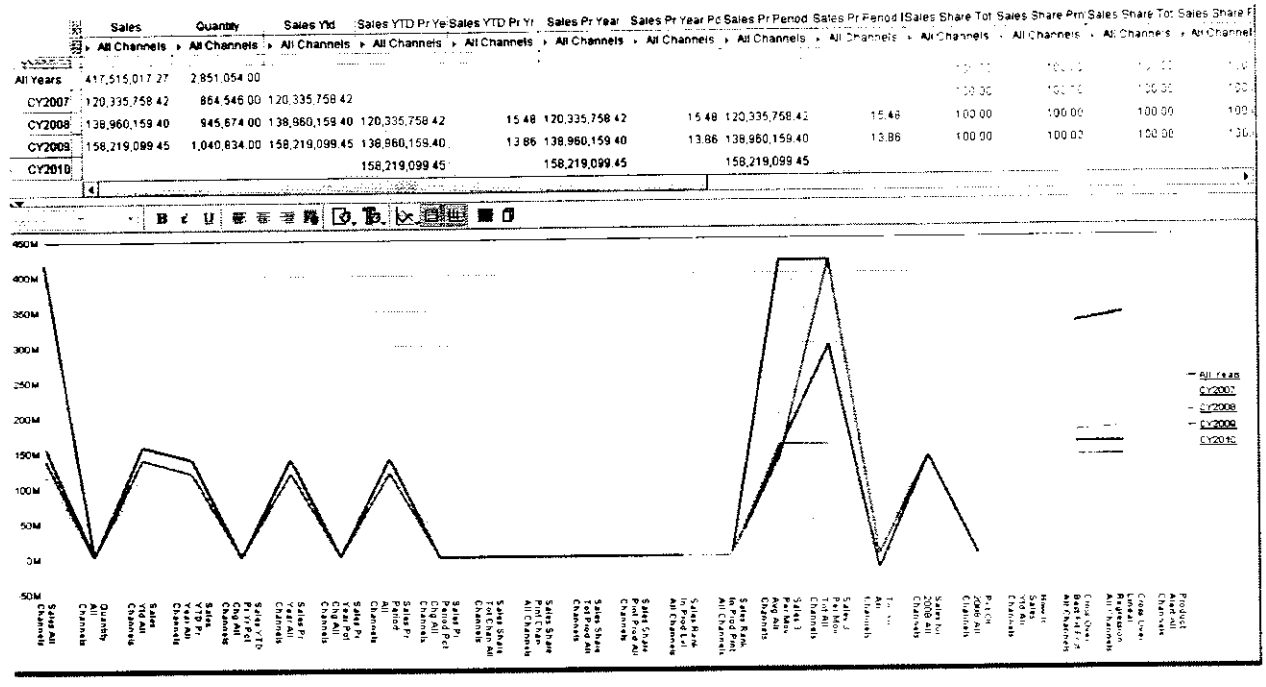


Fig. M Graphical Representation of query result on Expense_WH-Experiment 3



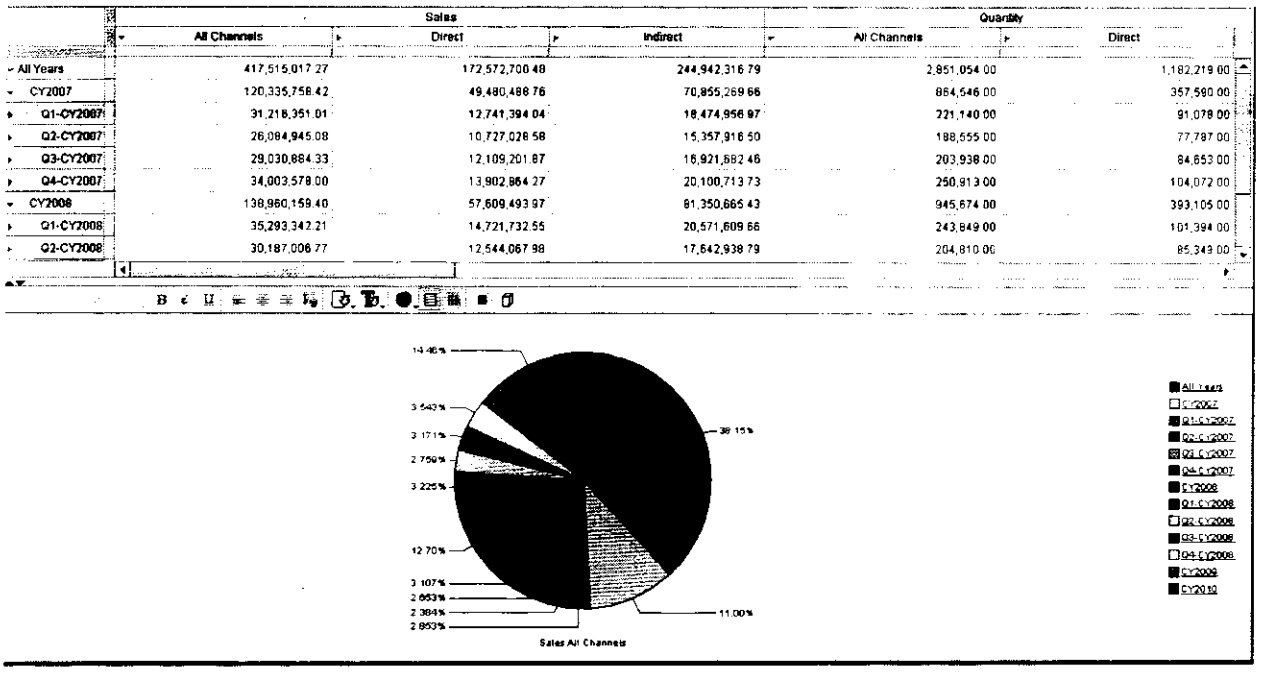
	Sales	Quantity	Sales Ytd	Sales YTD Pr Ye	Sales YTD Pr Yr	Sales Pr Year	Sales Pr Year Pc	Sales Pr Period	Sales Pr Period	Sales Share Tot	Sales Share Prn	Sales Share Tot	Sales Share F
	All Channels	All Channels	All Channels	All Channels	All Channels	All Channels	All Channels	All Channels	All Channels	All Channels	All Channels	All Channels	All Channel
All Years	417,515,017 27	2,851,054 00								100 00	100 00	100 00	100
CY2007	120,335,758 42	864,546 00	120,335,758 42							100 00	100 00	100 00	100
CY2008	138,960,159 40	945,674 00	138,960,159 40	120,335,758 42		15 48	120,335,758 42	15 48	120,335,758 42	100 00	100 00	100 00	100
CY2009	158,219,099 45	1,040,834 00	158,219,099 45	138,960,159 40		13 86	138,960,159 40	13 86	138,960,159 40	100 00	100 00	100 00	100
CY2010				158,219,099 45			158,219,099 45		158,219,099 45				

Fig. N Graphical Representation of query result on OLAPTRAIN-Experiment 2



	Sales	Quantity	Sales Ytd	Sales YTD Pr Ye	Sales YTD Pr Yr	Sales Pr Year	Sales Pr Year Pc	Sales Pr Period	Sales Pr Period	Sales Share Tot	Sales Share Prn	Sales Share Tot	Sales Share F
	All Channels	All Channels	All Channels	All Channels	All Channels	All Channels	All Channels	All Channels	All Channels	All Channels	All Channels	All Channels	All Channel
All Years	417,515,017 27	2,851,054 00								100 00	100 00	100 00	100
CY2007	120,335,758 42	864,546 00	120,335,758 42							100 00	100 00	100 00	100
CY2008	138,960,159 40	945,674 00	138,960,159 40	120,335,758 42		15 48	120,335,758 42	15 48	120,335,758 42	100 00	100 00	100 00	100
CY2009	158,219,099 45	1,040,834 00	158,219,099 45	138,960,159 40		13 86	138,960,159 40	13 86	138,960,159 40	100 00	100 00	100 00	100
CY2010				158,219,099 45			158,219,099 45		158,219,099 45				

Fig. O Graphical Representation of query result on OLAPTRAIN-Experiment 2



	Sales			Quantity	
	All Channels	Direct	Indirect	All Channels	Direct
All Years	417,515,017.27	172,572,700.48	244,942,316.79	2,851,054.00	1,182,219.00
CY2007	120,335,758.42	49,480,488.76	70,855,269.66	864,546.00	357,590.00
Q1-CY2007	31,218,351.01	12,741,394.04	18,474,956.97	221,140.00	91,078.00
Q2-CY2007	26,084,945.08	10,727,028.58	15,357,916.50	188,555.00	77,787.00
Q3-CY2007	29,030,884.33	12,109,201.87	16,921,682.46	203,938.00	84,653.00
Q4-CY2007	34,003,578.00	13,902,864.27	20,100,713.73	250,913.00	104,072.00
CY2008	138,960,159.40	57,609,493.97	81,350,665.43	945,674.00	393,105.00
Q1-CY2008	35,293,342.21	14,721,732.55	20,571,609.66	243,849.00	101,394.00
Q2-CY2008	30,187,006.77	12,544,067.98	17,642,938.79	204,810.00	85,349.00

Fig. P Graphical Representation of query result on OLAPTRAIN-Experiment 2

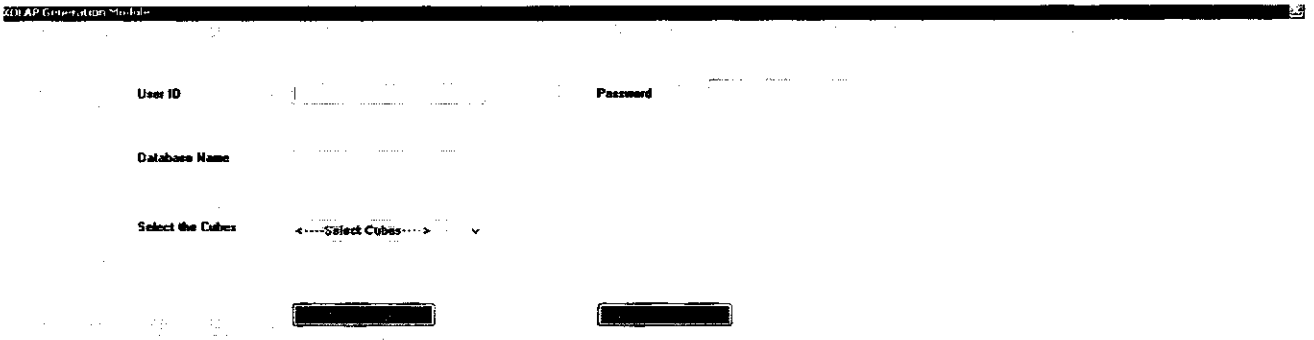


Fig. Q View of transformation Interface of RO LAP cubes into XOLAP