Kumaraguru College of Technology
Department of Computer Science and Engineering
Coimbatore– 641006.
April 2003

OLAP RIMS EXPRESS

Project work done at

P-1008

IVEGA CORPORATION

PROJECT REPORT

Submitted in partial fulfillment of the
Requirements for the award of the degree of

Master of Computer Applications
Bharathiar University, Coimbatore

Submitted by

PRASHANTH RAO M
Reg.No: 0038M1049

Internal Guide

Mr. A.Muthu Kumar M.C.A., M.Phil.,
Dept. of Computer Science & Engineering,
Kumaraguru College of Technology,
Coimbatore

External Guide

Mr.Raghunandan S
Senior Associate (Projects)
Ivega Corporation
Bangalore

CERTIFICATE

This is to certify that the project work entitled

OLAP RIMS EXPRESS

Submitted to the

Department of Computer Science and Engineering

Kumaraguru College of Technology

In partial fulfillment of the requirements for the award of the degree of Master of Computer Applications is a record of original work done by Prashanth Rao M. Reg.No.0038M1049 during his period of study in the Department of Computer Science and Engineering, Kumaraguru College of Technology, Coimbatore under my supervision and this project work has not formed the basis of award of any Degree/Diploma Associateship/Fellowship or similar title to any candidate of any university.

[illegible]
Professor and Head

[illegible]
Staff-in-charge

Submitted for University Examination held on [illegible]

[illegible]
Internal Examiner

[illegible]
External Examiner 16/4/03

Ivega

March 30, 2003

The Principal
Kumaraguru College of Technology
Coimbatore
641 006

Dear Sir,

This is to certify that **Mr. Prashanth Rao M** of **Kumaraguru College of Technology,** Coimbatore, has worked on the academic project entitled **'OLAP RIMS EXPRESS'** for Ivega Corporation.

The OLAP RIMS EXPRESS is a prototype model for the existing system. He has completed the project according to the specified architecture. He has also created an RDBMS model for testing the performance of the OLAP RIMS EXPRESS, which has proved that OLAP implementation of the existing system would yield a much better result.

We are very satisfied with the results of his work. His working attitude, enthusiasm and proficiency have been outstanding. He is good at OLAP services in SQL Server. In addition he is also good at design and implementation of Database.

He needs to improve his communication skills particularly the team interaction. In addition he needs to improve on analysis and go that extra mile and deliver, to add value to the work.

We wish him the very best in all his future endeavors.

For Ivega Corporation

for, Jyotsna Khuradi

Vasantha Nair
Manager-HR

DECLARATION

I here by declare that the project entitled **OLAP RIMS EXPRESS** – An OnLine Analytical Processing Research Information Management System Express, submitted to Bharathiar University as the project work of Master of Computer Application Degree, is a record of original work done by me under the supervision and guidance of **Mr.Raghunandan S, Senior Associate Projects Ivega Corporation**, Bangalore and **Mr.A.Muthu Kumar, M.C.A., M.Phil.,** Department of Computer Science, Kumaraguru College of Technology and this project work as not found the basis for the award of any Degree/Diploma/Associate-ship/Fellow-ship or similar title to any candidate of any University.

Place: COIMBATORE
Date: 08/04/2005

[illegible]
Signature of the Student

ACKNOWLEDGEMENT

Before I present this project, I wish to express my sincere gratitude to **Dr.K.K.Padmanabhan, Ph.D.,** Principal, Kumaraguru College of Technology, Coimbatore for his encouragement.

I wish to express my sincere thanks to the head of the department **Dr.S.Thangasamy, Ph.D.,** for having given me permission to carry out the project.

I also extend my heartfelt thanks to **Mr.Giri Devanur** Founder and Chief Executive Officer and **Mr.Raghupati Bhandi** Chief Operating Officer, Ivega Corporation for providing me the opportunity to do this project in this esteemed organization.

I also owe a deep sense of gratitude to **Mr.Raghunandan S** Senior Associate, Projects, **Mr.V.Vinay Kumar** Senior Associate and **Mrs.Priya Rao** Senior Associate for their Guidance, Constructive ideas and Constant Inspiration throughout the duration of the project.

Special thanks are due to **Mr.A.Muthu Kumar, M.C.A., M.Phil.,** my guide and tutor for his continued interest and guidance at all stages of this project.

Finally I thank all those who have contributed in various ways to this project by ways of their valuable suggestions and ideas.

SYNOPSIS

The project work entitled "**OLAP RIMS EXPRESS**"-An OnLine Analytical Processing Research Information Managament System is designed and developed at Ivega Corp. OLAP RIMS Express is an equity research and portfolio analysis platform that integrates internal and external stock and portfolio information.

Ivega Corp. has an impressive track record in executing projects in Chicago, Boston, Fort Lauderdale, New Jersey and Frankfurt with quality and client satisfaction as primary objectives. It has provided special expertise for developing many applications based systems.

OLAP RIMS EXPRESS has and continues to be a repository of in-house stock and portfolio information, covering analyst earnings estimates, target prices and valuations, portfolio manager and analyst votes, portfolio holdings, and in-house analyst notes and research. OLAP RIMS EXPRESS is also an open platform, allowing full integration of third party data including stock prices, fundamentals, street estimates, and benchmark constituents.

CONTENTS

1.1 PROJECT OVERVIEW

AIM OF THE PROJECT	:	TO DEVELOP THE PROTOTYPE MODEL OF OLAP RIMS EXPRESS.
DEVELOPED AT	:	IVEGA CORPORATION BANGALORE
PURPOSE	:	THE PURPOSE OF OLAP RIMS EXPRESS IS TO BUILD AN EQUITY RESEARCH AND PORTFOLIO ANALYSIS PLATFORM THAT IS IMPLEMENTED IN OLAP FOR ITS MULTIDIMENSIONAL PERSPECTIVE.
FRONT END	:	ASP, HTML
BACK END	:	MICROSOFT SQL SERVER 2000, MICROSOFT SQL SERVER 2000 ANALYSIS SERVICE
SUPPORT TOOL	:	MICROSOFT WORD 2000 FOR DOCUMENTATION, MICROSOFT POWERPOINT FOR PRESENTATION
HARDWARE SUPPORT	:	MICROSOFT WINDOWS BASED COMPAQ SYSTEM WITH INTEL PENTIUM 4 CPU 1.60 GHz, AT/AT 128 MB RAM, 37.2 GB HDD.
EXTERNAL GUIDE	:	Mr. RAGHUNANDAN S
INTERNAL GUIDE	:	Mr.A.MUTHU KUMAR
SUBMITTED TO	:	KUMARAGURU COLLGE OF TECHNOLOGY BHARATHIAR UNIVERSIRY COIMBATORE
DEVELOPED BY	:	PRASHANTH RAO M

1.1 ORGANIZATION PROFILE

ivega

A SEI-CMM LEVEL 4 AND AN ISO 9001:2000 SOFTWARE FIRM

Company profile

Ivega... our name aptly sums up what we stand for.

'I' is for Innovation. **'VEGA'**, taken from Sanskrit, means speed. We combine speed and innovation to deliver superior business and IT solutions to our clients.

Headquartered in Bangalore, the IT hub of India, we use an extensive state-of-the-art offshore development facility to deliver high quality and cost-effective solutions, catering to our clients' business requirements.

Mission

"To help our clients build superior processes and systems, to enhance their performance, with our expert teams and their exceptional ideas."

Core principles

"We Listen to our Clients and deliver high **Quality** products and **Reliable** services by constant **Innovation** through **Adaptive** processes while maintaining a **Transparent** approach".

ivega

- ✓ One of the youngest Indian companies to have achieved the ISO 9001 certification and SEI CMM Level 4.
- ✓ Our clients' are assured that quality processes are incorporated in all projects, facilitating on-time and on-budget delivery.

- ✓ Contractually commit to maintaining agreed upon attrition levels of key resources, with attached penalties
- ✓ Provide a ninety-day warranty for fixing defects, free of cost.
- ✓ Ivega Idea Lab at Bangalore HQ has produced revolutionary products and ideas that have significantly enhanced our project execution processes.
- ✓ Strongly believes in joint development, involving the client in all the aspects of a project.
- ✓ Using I-PRO Ivega's web-enabled project management collaboration and knowledge management tool that manages and tracks each and every phase of a project life cycle from launch to release, and aids resource management, issue management and escalation, support opportunities, and service delivery.

Partners

We have invested in developing alliances with key business and technology partners to deliver added value to our clients.

Ilantus Technologies - With offices across the globe in US, Singapore and India, iLantus has global experience in consulting, technology integration, software development and maintenance of Identity Management solutions.

Infrasoft Technologies - Banking, financial services, insurance being the core focus industries, Infrasoft specialises in providing n-tier enterprise banking application and software solutions in the areas of Wealth Management, Trading Systems, Bank Assurance and eCommerce Solutions.

Catalyst Infotech - Catalyst is an integrated products and services firm specializing in web and wireless technologies for the financial services vertical that includes securities trading, banking and the mutual fund industries

Teksels S.A - Ivega and Teksels work in tandem offering global / local service through our state-of-the-art software engineers, using the on-site / offshore model.

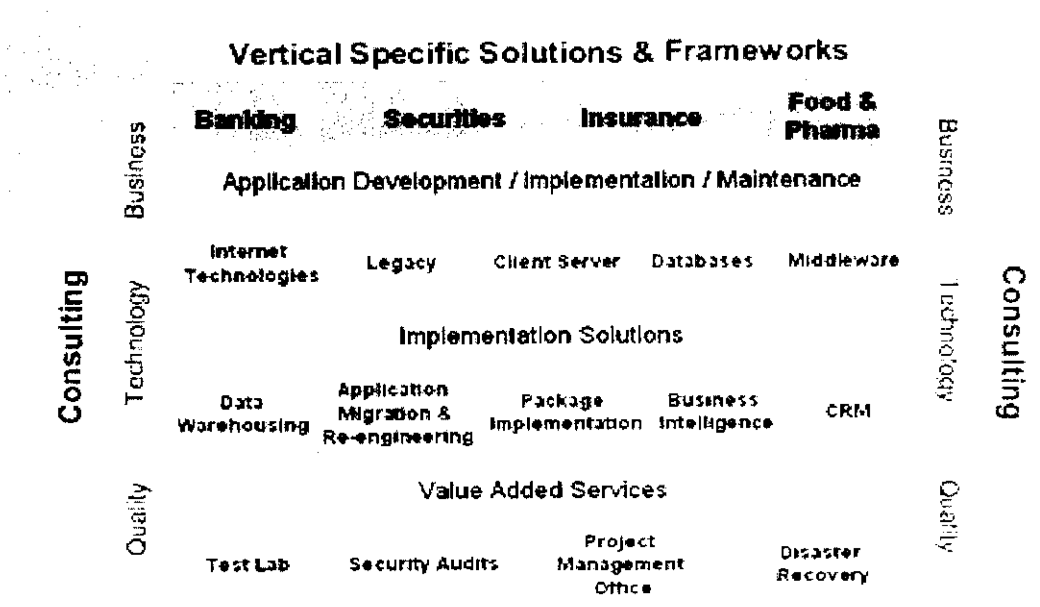
NueVista Group - NueVista Group is an IT services firm with expertise in quality assurance and testing, application development and integration, application maintenance, network services and corporate governance.

Computer Software Services, LLC (CSS) - CSS is a leading provider of Information Technology solutions for Atlanta and the Southeast region.

Our partnerships with **IBM** and **BEA Systems** provide us access to their latest technologies facilitating the execution of high-end solutions for our clients.

Services

Ivega offers a wide range of services from custom software development, maintenance & re-engineering services, large application development, enterprise integration services to establishing a dedicated offshore development center. The following matrix best describes Ivega's service portfolio:



Industry

Converging vast domain knowledge and in-depth technology expertise, we have put together a number of industry- specific core teams. We add significant value to our client's business, by developing innovative solutions to address the key challenges facing our focus industries, worldwide.

We leverage our extensive specialist expertise, CMM Level 4 processes and effective delivery models to provide a range of services to our clients in the Banking, Securities, Insurance, Health Care and other industries.

Facility

Located in Bangalore, India, we utilize an extensive offshore software development centre (ODC) spanning 35,000 Sq. ft. capable of housing 350+ software consultants with all necessary hardware and software facilities. We are situated on a business park in the outskirts of Bangalore. Surrounded by palm trees and vast lawns, Ivegans enjoy an excellent working environment.

The ODC is currently supporting over 25 offsite engagements for clients across the world. It houses state-of-the-art technological infrastructure, which includes:

- ✓ A network of 350+ workstations. Several Netware, UNIX and WINDOWS NT servers
- ✓ 24/7 connectivity through 512 Kbps dedicated line
    - Backup Link via ISDN Line as secondary
- ✓ 1Gbps Switched 400 Nodes network
    - Network secured
    - Firewall Checkpoint
    - Cisco PIX Router
    - Anitivirus

- ✓ Automated Backup and Restore Systems
- ✓ Redundant data back ups and archives for COB.

Consequently, Ivega is able to provide full project management support in each of its services while delivering high quality and cost-effective services to each of its clients.

Brief history

1997: Incorporation, with 10 employees in the first year

1998: Achieved ISO 9001 certification

1999:

- Opened San Francisco office
- Growth in number of employees to 100
- Raised 3 rounds of venture funding

2000: Achieved CMM Level 4 Certification

2001:

- Started operations in Europe
- Growth in number of employees to 150

2002:

- ISO certification updated to ISO 9001:2000 series
- Opened offices in Chicago, Boston, Fort Lauderdale, New Jersey and Frankfurt

2.1 FEASIBILITY STUDY

The preliminary investigations of the system examine the project feasibility, i.e. the likelihood of the project being useful to the organization.

Three specific tests were carried out on the system to project its feasibility namely, Operational, Technical and Financial/Economical.

Operational Feasibility

Any project proposed can be beneficial only if it satisfies the organizational requirements. In any mature MIS setup, a system not only needs to be robust but also needs to communicate and work in tandem with the other existing support systems. Following are some points underlining the operational feasibility of the system.

The system was well supported by the management with the MIS manager taking personal interest in the system development process. The current system though functional in practice needs a lot of trimming and streamlining.

The clear advantage of using better software prevented user resistance if any from disrupting the development process. The users were involved with the system right from the beginning and were always in touch with the latest developments.

The proposed system makes a best effort to satisfy the requirements of the user, keeping in mind certain infrastructure constraints.

Since the most trivial issues assume a major problematic state later in the development cycle, every possible aspect of operational feasibility was checked.

Technical Feasibility

The technical issues generally raised during the investigation are discussed below. The organization has a very well integrated MIS department with the requisite technology to cater to the needs of the proposed system. Any additional requirements can also be satisfied.

The proposed technology has the capacity to hold the data required to use the new system.

The system is very much open by nature and can be easily expanded in the near future to satisfy newly emerging technology.

The usage of a reliable RDBMS like MSSQL along with stringent design coding standard followed guarantees accuracy, reliability, ease of data access and data security.

Financial and Economical Feasibility

The cost involved in designing and developing a system should be a good investment for the organization. The financial benefits must equal or exceed the costs. The financial and economic issues raised during preliminary investigations are answered below.

- The cost of Hardware and software for the application already exists.
- The use of existing software and developing a new application using it only increases the gain.
- Benefits accrue through quicker solving of queries resulting in timely submission of reports.
- The cost if nothing were to change from the present system, wouldn't necessarily increase but the gain from the system stagnates or decreases due to poor handling of the existing system.
- The proposed project passed all the tests and was declared feasible to the organization and it's functioning.

2.2 PRODUCT FUNCTIONS

RIMS Express Overview

RIMS Express ("**R**esearch **I**nformation **M**anagement **S**ystem") is the third-generation of RIMS products dating back to 1975. Designed and developed at Ivega Corporation. RIMS Express is an equity research and portfolio analysis platform that seamlessly integrates internal and external stock and portfolio information.

First and foremost, RIMS has and continues to be a repository of in-house stock and portfolio information, covering analyst earnings estimates, target prices and valuations. portfolio manager and analyst votes, portfolio holdings, and in-house analyst notes and research. RIMS is also an open platform, allowing full integration of third party data including stock prices, fundamentals, street estimates, and benchmark constituents. RIMS provides a standard set of stock and portfolio functions including stock valuation, sorting and screening, portfolio against benchmark characteristics and sector weights, portfolio contribution, and multiple-portfolio holdings by sector reports. Finally, RIMS produces overnight batch updating and overnight report generation for portfolio managers and analysts.

Site Structure

The RIMS Express site structure can be described as three primary and three secondary sections, each with a set of pages dedicated to either specific data or a specific analytical function.

Stocks—This primary section contains pages dedicated to individual stocks: Summary, Reference, House Research, House Valuation, House Earnings, Dividends, Votes, CM Holdings, Street Research, Street Estimates, Street Recommendations, Street Consensus, EPS Surprise, EPS Revision, Strength, Insider Trends, Price Ratios, Fundamentals, Price Chart, and Performance.

Portfolios—This primary section contains pages dedicated to individual portfolios: Portfolio Overview, Reference, Characteristics Profile, Characteristics Scatterplot, Weight by Holdings, Weight by Sectors, Daily Performance, Daily Contribution, Daily Active Contribution, Monthly Active Contribution, AggMan, User Models, and Adjusted P/Es.

Reports—This primary section contains pages that provide access to key tools and specific stock or portfolio reports—"Sort & Screen" Tool, Weight Range Tool, Portfolio Deviation Tool, Overnight Reports, Price Reports, Afternoon Performance Reports, Webbook Reports, Manager Aggregates Deviation Reports, PMT Aggregates Deviation Reports, and Industry Deviation Reports.

Analysts—This secondary section contains a page to view House Research by Analyst instead of by Stock, and a page with the Analyst's Biography.

Home—This section contains pages that provide administrative functions, including data integrity—Home, Login, Personalize, Price Update Status, Holdings Update Status, Analyst's Changes, Analyst Initials Maintenance, Industry Classification Maintenance, Benchmark Exception Report, Industry Exception Report, Report Maintenance.

Site Map—This section contains a site map with links to all other pages.

Data Types

RIMS Express integrates investment data from many sources—both internal and external to the firm. The various data elements cover security classification, analyst-generated research, holdings, indexes, company financial statements, broker estimates, and many more data elements. Internally, the following source types are provided:

CM Stock Classification Data—CM new global sector, industry, and subindustry classification; and CM analyst team sectors and analyst initials. This information is stored and maintained in RIMS Express.

CM Analyst–Generated Stock Data—Earnings estimates in local currency, valuation parameters (including target prices, upsides, and risk-adjusted returns), votes, and forecasted long-term growth rates. This information is stored and maintained in RIMS Express.

CM Analyst–Generated Stock Research—Future plans for delivery of written research including attachments (text, documents, spreadsheets, etc.). This information would be stored in the IBES Trapeze module of RIMS Express.

CM Portfolio Identification Data—Portfolio longname, portfolio shortname, the portfolio manager, client restrictions, and primary benchmark. Much of this information is either stored and maintained in CMdb (for portfolios) or stored and maintained in RIMS Express (for models and aggregates).

CM Portfolio Holdings Data—Stocks held, shares of stocks held, cost basis of stocks held, cash held. Much of this information is either stored and maintained in CMdb (for portfolios) or stored and maintained in RIMS Express (for models and aggregates).

CM Analyst Data—Analyst name, and biography. This information is stored and maintained in Web books.

Externally, data is primarily sourced from these vendors: Bloomberg, Extel, and Muller for prices; Worldscope and Compustat for fundamentals; IBES for street estimates and recommendations; and MSCI, S&P, FTSE, and Russell for benchmark constituents. All external data are updated daily. Together these data vendors provide these data source types:

Stock Classification Data—Cusip, sedol, local ticker, name, total shares outstanding, class shares outstanding, country, currency, stock exchange listing, domicile, headquarters, etc.

Street Estimates and Recommendations—EPS estimates (FY1, FY2, FY3, and quarters), long-term EPS growth estimates, cash flow estimates, sales estimates, dividend estimates.

Stock Financial Statement Data—reported earnings, revenues, expenses, assets, liabilities, equity (book value), ROE, ROA, etc.

Stock Prices—local prices and FX rates. (All data, including prices, are updated daily—prices are neither real-time nor 15-minute delayed.)

Indexes—Index constituents: shares held and stock weights.

Technology Architecture

RIMS is supported by an underlying platform and n-tiered system architecture licensed from IBES International, Inc. (a division of Primark) optimized for the development of a browser-based application. The underlying platform's relational database has allowed the seamless integration of different data types. The n-tiered architecture includes a sophisticated metadata layer, which effectively organizes and generates analytics to facilitate system efficiency, maintenance, and future development. The User Interface Component of RIMS Express was developed using industry standard technologies such as DHTML (Dynamic HTML), JavaScript and lightweight binary components (ActiveX Controls). Early on in the design of RIMS Express, the System Architect felt very strongly that the final product needed to have the following attributes:

A rich Graphical User Interface, of the kind to which users have grown accustomed

The ability of the application to "self-deploy", and to update itself as new functionality became available in the system

The use of industry-standard technologies, so development resources (developers, training and knowledge base) would be readily available

These requirements posted quite a challenge, as traditionally, System Architects have been forced to choose between either the two-tier, binary-centric, fat-client application or the ultra-thin web page oriented architecture when deciding on the appropriate architecture for application development. As history has proven, neither architecture is totally satisfying, given the fact that the two-tier client/server architecture is simply too costly to maintain, and the ultra-thin client too simplistic in its functionality. The RIMS Express architecture gets around these issues by utilizing the powerful DHTML technology, which gives the user fast, responsive interaction with the application. However, due to the newness of the technology, there are instances where this technology has fallen short, e.g. when displaying large sum of tabular data. During these situations, the RIMS Express development team supplemented the DHTML with appropriate lightweight binary controls, which are also capable of self-deployment. The end result is a rich, fast and functional browser-based application with all of the modern-age niceties such as charts, scatterplots and sophisticated report generation, packaged in a totally self-deployable environment. With this architecture, we think we will able to minimize the Total Cost of Ownership (TCO) while providing a development structure that will keep up with the ever-changing business environment.

2.3 EXISTING SYSTEM

A "Browser-based" Architecture

As the name implies, a Browser-based architecture is one in which a Web-based application such as RIMS Express is deployed via an Internet Browser environment (in the case of RIMS Express, it is specifically targeted for Microsoft Internet Explorer version 5 and above). Prior to the rising preeminence of the Internet Browser as a viable deployment platform for applications, system designers and architects were confined to the traditional Windows-based Client/Server model as the predominant architecture. These applications consist of a client component that runs on the end-user's workstation running Windows, and a relational database management system such as SQL Server or Oracle running on a shared database server. But as history has proven, this model is not sustainable over time as problems with version control of dynamic load libraries (DLLs) within the Windows environment are simply too delicate for most IT organizations to support. As the Web became more and more popular, developers were particularly enthusiastic over the concept of "thin client" or "ultra thin client" applications, as they later became known. The sole requirement for applications running under this architecture is for client workstation be able to run an Internet Browser type application, e.g. Netscape or Microsoft, and all of the logic for the system is presented to the user via web pages formatted using HTML.

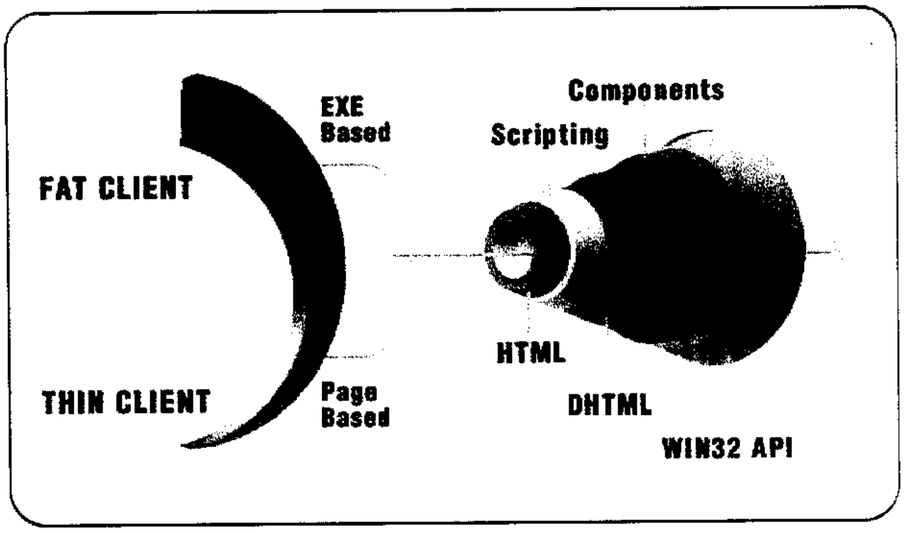
Early industry attempts at dynamic web applications were based on what is called server-side scripting, in which a client's interaction with the browser initiated the execution of a script up on the web server itself (Microsoft's ASP, or Active Server Pages, is an example of this architecture). This script in turn created a dynamic, new version of a page every time it was requested. Though this is still a viable architecture in some situations, it becomes considerably less attractive in applications where subsequent data manipulation on the client workstation is desired. For example, to request that a matrix of data on a page be sorted in a different order from that in which it was initially retrieved, a server-side application needs to re-retrieve, re-sort, and re-send the entire data set back to the client. This full-roundtrip requirement causes an unnecessary drain on both server resources and network bandwidth, not to mention the perceived delay on the part of the user.

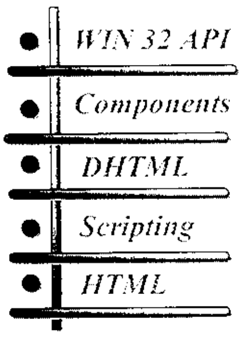
As the browser became more powerful, another option became available. IE version 4.0 allowed client-side scripts (whereby application logics can be embedded as part of a HTML page) to be written that could retrieve data, format and display it on the page, and allow it to be manipulated on the user's workstation without the necessity of making additional, redundant server calls for the already retrieved data. These client-side script executions and data manipulations also more fully utilized the computational power of the workstation, and achieved a much better balance of client and server resources. And because the pages resided on the server, the deployability headaches of fat-client applications were relieved. Finally, systems became more scalability as a result of the inherent "stateless" nature of web-based applications. Unlike fat-client applications, web-based applications don't require persistent database connections, so systems could now scale to thousands of users or more using equivalent resources.

RIMS Express takes this thin-client architecture one step further, by using specially written client-side binary components packaged as a dynamically downloadable COM Object to enhance and optimize the data retrieval process. Sophisticated caching algorithms, security, and business logic reside in these components to provide enhanced capabilities such as pivoting, formatting, and aggregation that it was found were too resource-intensive to accomplish in scripting alone.

If Web-based techniques are to be a core paradigm for building applications, both browsers and servers must provide advanced features and robust performance. MS version of DTHML adheres closely to the W3C document object model (OM). Introduced with version 4.0 of their Web browser, DHTML provides powerful layout and presentation features. In addition, its event model allows Web pages to be highly interactive. DHTML permits a level of interactivity such as is seen in Java applets with performance typical of in-page scripts. Future versions of the Web browsing client will improve the performance of interface elements to achieve results closer to that experienced with applications written with compiled languages.

Scriptlets are reusable chunks of DHTML functionality formed by combining browser-based scripting with the COM component software technology provided by Windows.





Rich application programming interface for direct access to all system capabilities.

Robust mechanism for applications to expose functionality and interfaces.

Advanced document object model that provides greater control and interactivity.

Simplified programming language interface that is easily ulitized by web applications.

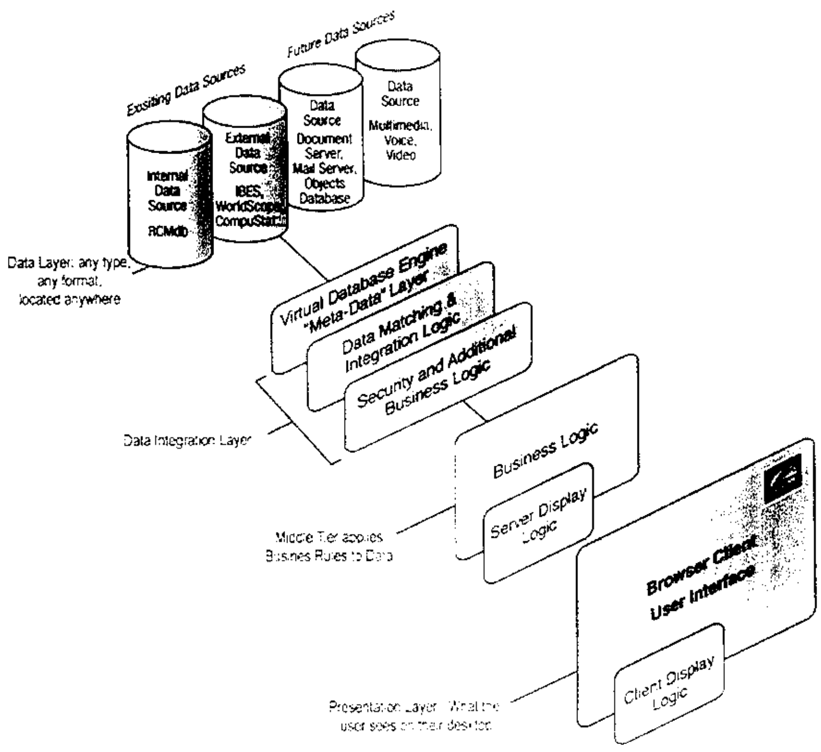
Basic method for rendering web applications across a wide variety of platforms.

High Level Architecture Overview

There are 4 major components within the RIMS Express system architecture:

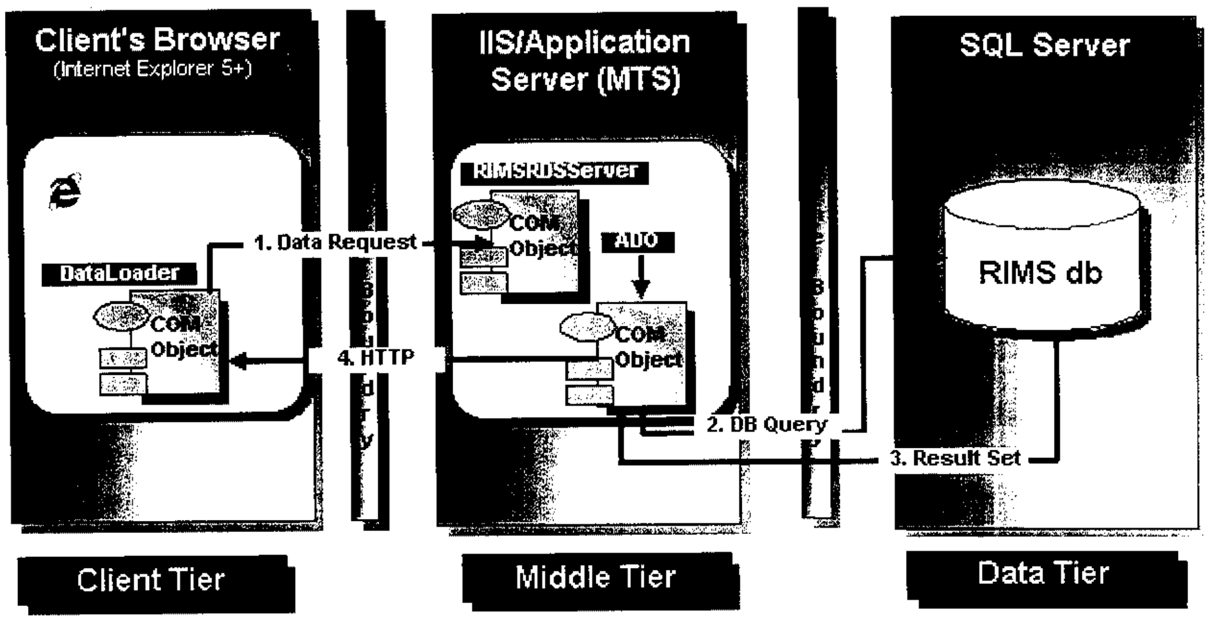
- **Database Management** – Powerful relational Database Engine to provide quick responsive data access.
- **Data Integration** – System logic to merge and cleanse the vast amount of data that RIMS Express uses as its sources
- **Business Logic** – Application logic implementing business rules specific to the client, specified for the most part by staff Quantitative and Research Analysts

- **Presentation** – Rich user interface conforming to latest trends in web design, including dynamic menus/dropdowns/listboxes, clickable column headings for sorting, mouse rollovers, right-click context menus, drag-and-drop, etc.


Existing Data Sources
Future Data Sources
Internal Data Source RCMdb
External Data Source IBES, WorldScope, CompuStat
Data Source Document Server, Mail Server, Objects Database
Data Source Multimedia, Voice, Video
Data Layer: any type, any format, located anywhere
Virtual Database Engine "Meta-Data" Layer
Data Matching & Integration Logic
Security and Additional Business Logic
Data Integration Layer
Business Logic
Server Display Logic
Middle Tier applies Busines Rules to Data
Browser Client User Interface
Client Display Logic
Presentation Layer: What the user sees on their desktop

Data Transport Technology

Among the new technologies used by RIMS Express, one of the most significant is RDS. RDS stands for Remote Data Services, and as the name implies. RDS supplies a number of services around data acquisition and manipulation. Without getting too technical, the two primary RDS services we use in RIMS Express are the Advanced Data Control, which is an invisible ActiveX component embedded within a web page that allows the page to communicate with an external datasource through any web server of choice. This encapsulates the origin of the data from the client's point of view, which in turn greatly facilitates future changes to the underlying data source. The other service that is heavily leveraged within the RIMS Express environment is "disconnected recordsets", which facilitate the client-side data manipulation whereby the user can sort, filter and print data locally without ever having to re-request the data from the server. With this architecture, RIMS Express is able to provide the flexibility to seamlessly integrate any type of content from almost any database, with complete location transparency.



Result

The fundamental design goal for RIMS Express is to be the de-facto equity research and portfolio analysis repository. The application gathers information from multiple sources, both internal and external, then stores and categorizes it. With this information, different analytical views can be presented appropriate for the Analyst, regardless of where it is stored. RIMS Express has made it easier for users to access a much greater variety of data in a more timely fashion - no matter where it lives. This seemless integrated viewing of data at the desktop level dramatically empowers the analysts and Portfolio Managers. As financial data becomes increasingly abundant, the underlying system architecture for the application must be robust enough to handle not just what is available today, but rather factor in future information growth along with the additional data media options such as voice and video. RIMS Express is designed with the intent of accommodating this future growth.

2.4 REQUIREMENTS OF THE NEW SYSTEM

- A tool that provides the business analyst with a million spreadsheets at a time.
- The spreadsheets must be in a logical, hierarchical structure.
- The analyst must be able to move to a higher or lower level of detail or look at the data from additional perspectives.
- Browsing must be consistently fast.
- The application should contain analytical tools.
- The application must be able to handle the security requirements of sharing confidential information.
- In short the tool must be a "Multidimensional, Multilevel, Multiperspective analytical software tool".

2.5 PROPOSED SYSTEM

OLAP RIMS Express ("**R**esearch **I**nformation **M**anagement **S**ystem") is a prototype model. Designed and developed at Ivega Corp. OLAP RIMS Express is an equity research and portfolio analysis platform that integrates internal and external stock and portfolio information.

The mission statement of the OLAP RIMS Express is to retrieve the in-house and the street analyst data with the major pillar data's. OLAP RIMS Express is intended to describe and explain to the RIMS community how these pillars of analyst data and the Stock Valuation data are used within RIMS and the equity investment process.

OLAP RIMS has to be a repository of in-house stock and portfolio information, covering analyst earnings estimates, target prices and valuations, portfolio manager and analyst votes, portfolio holdings, and in-house analyst notes and research. OLAP RIMS allows full integration of third party data including stock prices, fundamentals, street estimates, and benchmark constituents. OLAP RIMS provides a standard set of stock and portfolio functions including stock valuation, portfolio contribution, and multiple-portfolio holdings by sector reports. Finally, OLAP RIMS produces overnight batch updating and overnight report generation for portfolio managers and analysts.

The OLAP RIMS Express provides an initial view for a portfolio manager or an analyst to see a summary of key data on a particular stock. On the page, the row, column contain key data items from dimension pages including:

Valuation—target price, 12-month forecasted upside return, risk-adjusted return:

Votes—of key Portfolio Management Team members and the analyst covering the stock;

Strength—performance of the stock for various time periods;

Holdings—CM product groupings and key benchmarks

Reference—market capitalization, shares outstanding, and sector/industry classification;

Earnings—EPS and P/E for last fiscal year, current fiscal year, and next 2 fiscal years.

Timeliness has been defined, as the output to the page should be displayed within few Seconds of the request being approved from the dimension page. The response time will also depend on the number of dimensions the data is viewed. To reduce the response time special storage structure has to be deployed.

Accuracy is focused on all data and fields within the system. The data that is present in the database should match with the olap report. A comparison between the database model report and the olap model report should be same and accurate.

The system **updates** the data at the end of every month. The DTS package is responsible for this work.

Reporting is the major work of olap and it is available on demand. The report shows a two dimensional view of the data. The members of one or more dimension appear in the column and row.

2.6 USER CHARACTERISTICS

- The user is required to have knowledge about MS OLAP to interact with the user interface screens developed using ASP.
- There are lists of levels as per the user requirements. These levels make sure that unauthorized users are restricted.
- Viewing specific hierarchy is classified as per the levels.
- There is an administrator who has all the privileges. Access to each user is restricted by means of access roles.

3.1 HARDWARE CONFIGURATION

- MICROSOFT WINDOWS BASED COMPAQ SYSTEM
- INTEL PENTIUM 4 CPU 1.60 GHz
- AT/AT 128 MB RAM
- 37.2 GB HDD
- COMPAQ 5500 COLOR MONITOR
- COMPAQ DVD/CD ROM DRIVE
- HP LASERJET 4050 SERIES PS PRINTER
- 1.44 MB FDD

3.2 SOFTWARE CONFIGURATION

- MICROSOFT WINDOWS 2000 SERVICE PACK 2
- MICROSOFT INTERNET EXPLORER 6
- MICROSOFT OFFICE 2000 PROFESSIONAL

RDBMS

- MICROSOFT SQL SERVER 2000
- MICROSOFT SQL SERVER 2000 ANALYSIS SERVICE

TOOLS

- MICROSOFT VISUAL STUDIO 6.0 ENTERPRICE EDITION
- SEGATE CRYSTAL REPORT DEVELOPER EDITION

Introduction to RDBMS

Physical Database Design

The I/O subsystem (storage engine) is a key component of any relational database. A successful database implementation usually requires careful planning at the early stages of a project. The storage engine of a relational database requires much of this planning, which includes determining:

- What type of disk hardware to use, such as RAID (redundant array of independent disks) devices?
- How to place the data onto the disks?

RAID

RAID (redundant array of independent disks) is a disk system that comprises multiple disk drives (an array) to provide higher performance, reliability, storage capacity, and lower cost. Fault-tolerant arrays are categorized in six RAID levels, 0 through 5. Each level uses a different algorithm to implement fault tolerance.

Although RAID is not a part of Microsoft® SQL Server™ 2000, its implementation can directly affect the way SQL Server performs. RAID levels 0, 1, and 5 are typically used with SQL Server.

RAID is available only on Microsoft Windows NT 4.0 and Microsoft Windows 2000.

A hardware disk array improves I/O performance because I/O functions, such as striping and mirroring, are handled efficiently in firmware. Conversely, an operating system based RAID offers lower cost but consumes processor cycles. When cost is a consideration and redundancy and high performance are required, Microsoft Windows® NT® stripe sets with parity or Windows 2000 RAID-5 volumes are a good solution.

Data striping (RAID 0) is the RAID configuration with the highest performance, but if one disk fails, all the data on the stripe set becomes inaccessible. A common installation technique for relational database management systems is to configure the database on a RAID 0 drive and then place the transaction log on a mirrored drive (RAID 1). One can get the best disk I/O performance for the database and maintain data recoverability (assuming one performs regular database backups) through a mirrored transaction log.

If data must be quickly recoverable, consider mirroring the transaction log and placing the database on a RAID 5 disk. RAID 5 provides redundancy of all data on the array, allowing a single disk to fail and be replaced in most cases without system downtime. RAID 5 offers lower performance than RAID 0 or RAID 1 but higher reliability and faster recovery.

Data Placement Using Filegroups

Microsoft® SQL Server™ 2000 allows one to create tables or indexes on a specific filegroup within the database, rather than across all filegroups in a database. By creating a filegroup on a specific disk or RAID (redundant array of independent disks) device, one can control where tables and indexes in the database are physically located. Reasons for placing tables and indexes on specific disks include:

- Improved query performance.
- Parallel queries.

Files and Filegroups

Microsoft® SQL Server™ 2000 maps a database using a set of operating system files. All data and objects in the database, such as tables, stored procedures, triggers, and views, are stored within these operating-system files:

Primary - This file contains the startup information for the database and is used to store data. Every database has one primary data file.

Secondary - These files hold all of the data that does not fit in the primary data file. If the primary file can hold all of the data in the database, databases do not need to have secondary data files. Some databases may be large enough to need multiple secondary data files or to use secondary files on separate disk drives to spread data across multiple disks.

Transaction Log - These files hold the log information used to recover the database. There must be at least one log file for each database.

Logical Database Design

Using Microsoft® SQL Server™ 2000 effectively begins with normalized database design. Normalization is the process of removing redundancies from the data. For example, when one converts from an indexed sequence access method (ISAM) style application, normalization often involves breaking data in a single file into two or more logical tables in a relational database. Transact-SQL queries then recombine the table data by using relational join operations. By avoiding the need to update the same data in multiple places, normalization improves the efficiency of an application and reduces the opportunities for introducing errors due to inconsistent data.

However, there are tradeoffs to normalization. A database that is used primarily for decision support (as opposed to update-intensive transaction processing) may not have redundant updates and may be more understandable and efficient for queries if the design is not fully normalized. Nevertheless, data that is not normalized is a more common design problem in database applications than over-normalized data. Starting with a normalized design, and then selectively denormalizing tables for specific reasons, is a good strategy.

Whatever the database design, one should take advantage of these features in SQL Server to automatically maintain the integrity of the data:

CHECK constraints ensure that column values are valid.

DEFAULT and NOT NULL constraints avoid the complexities (and opportunities for hidden application bugs) caused by missing column values.

PRIMARY KEY and UNIQUE constraints enforce the uniqueness of rows (and implicitly create an index to do so).

FOREIGN KEY constraints ensure that rows in dependent tables always have a matching master record.

IDENTITY columns efficiently generate unique row identifiers.

timestamp columns ensure efficient concurrency checking between multiple-user updates.

User-defined data types ensure consistency of column definitions across the database.

By taking advantage of these features, one can make the data rules visible to all users of the database, rather than hiding them in application logic. These server-enforced rules help avoid errors in the data that can arise from incomplete enforcement of integrity rules by the application itself. Using these facilities also ensures that data integrity is enforced as efficiently as possible.

3.2.1 MICROSOFT SQL SERVER 2000

SQL Server 2000 provides two fundamental services to applications in a Windows® DNA environment:

- **The SQL Server 2000 relational database engine** is a modern, highly scalable, highly reliable engine for storing data. The database engine stores data in tables. Applications can submit Structured Query Language (SQL) statements to the database engine, which returns the results to the application in the form of a tabular result set. The specific dialect of SQL supported by SQL Server is called Transact-SQL. Applications can also submit either SQL statements or XPath queries and request that the database engine return the results in the form of an XML document.

The relational database engine is highly scalable. The SQL Server 2000. Enterprise Edition can support groups of database servers that cooperate to form terabyte-sized databases accessed by thousands of users at the same time. The engine is capable of handling the traffic of any Web site in the world. The database engine also tunes itself, dynamically acquiring resources as more users connect to the database, and then freeing the resources as the users log off. This means that the smaller editions of SQL Server can be used for individuals or small workgroups that do not have dedicated database administrators. Even large Enterprise Edition database servers running in production are easy to administer using the graphical user interface (GUI) administration utilities that are a part of the product.

The relational database engine is highly reliable and capable of running for long periods without down time. Administrative actions that required stopping and starting in earlier versions of the database engine can now be performed while the engine is running, increasing availability. The integration of the database engine with Windows 2000 and Windows NT® failover clustering allows one to define virtual servers that keep running even if one of the physical servers in the node fails. Where appropriate, log shipping can be used to maintain a warm standby server that can replace a production server within minutes of a failure.

The relational database engine is also highly secure. Login authentication can be integrated with Windows Authentication, so that no passwords are stored in SQL Server or sent across the network where they could be read by network sniffers. Sites can set up C2-level auditing of all users accessing a database, and can use Secure Sockets Layer (SSL) encryption to encrypt all data transferred between applications and the database.

The distributed query feature of the database engine allows one to access data from any source of data that can be accessed using OLE DB. The tables of the remote OLE DB data source can be referenced in Transact-SQL statements just like tables that actually reside in a SQL Server database.

The relational database engine is capable of storing detailed records of all the transactions generated by the top online transaction processing (OLTP) systems. The database engine can also support the demanding processing requirements for fact tables and dimension tables in the largest online analytical (OLAP) data warehouses.

- **Microsoft SQL Server 2000 Analysis Services** provides tools for analyzing the data stored in data warehouses and data marts. Certain analytical processes, such as getting a summary of the monthly sales by product of all the stores in a district, take a long time if run against all the detail records of an OLTP system. To speed up these types of analytical processes, data from an OLTP system is periodically summarized and stored in fact and dimension tables in a data warehouse or data mart. Analysis Services presents the data from these fact and dimension tables as multidimensional cubes that can be analyzed for trends and other information that is important for planning future work. Processing OLAP queries on multidimensional Analysis Services cubes is substantially faster than attempting the same queries on the detail data recorded in OLTP databases.

Application Support

Both the relational database engine and Analysis Services provide native support for the common Windows DNA or Win32 data access interfaces, such as ActiveX® Data Objects (ADO), OLE DB, and Open Database Connectivity (ODBC). Applications can use any of these application-programming interfaces (APIs) to send SQL or XML statements to the relational database engine using a native OLE DB provider or ODBC driver. SQL Server 2000 also introduces the ability to use HTTP to send SQL or XML statements to the relational database engine. Applications can use the multidimensional extensions of either ADO or OLE DB to send Multidimensional Expressions (MDX) queries to Analysis Services. Because SQL Server uses the standard Windows DNA data access APIs, the development of SQL Server applications is well supported by the

Microsoft application development environments. In addition, interactive query tools, such as Query Analyzer, provide templates, interactive debuggers, and interactive test environments that speed the ability of the programmers to deliver SQL Server applications.

In addition to supporting the data storage and OLAP processing needs of applications, SQL Server 2000 provides a full set of easy to use, graphical administration tools and wizards for creating, configuring, and maintaining databases, data warehouses, and data marts. SQL Server also documents the administration APIs used by the SQL Server tools, giving the ability to incorporate SQL Server administration functionality directly into the applications. The SQL Server administration APIs include:

SQL Distributed Management Objects (SQL-DMO), a set of COM objects that encapsulates the administration functions for all of the entities in the relational database engine and databases.

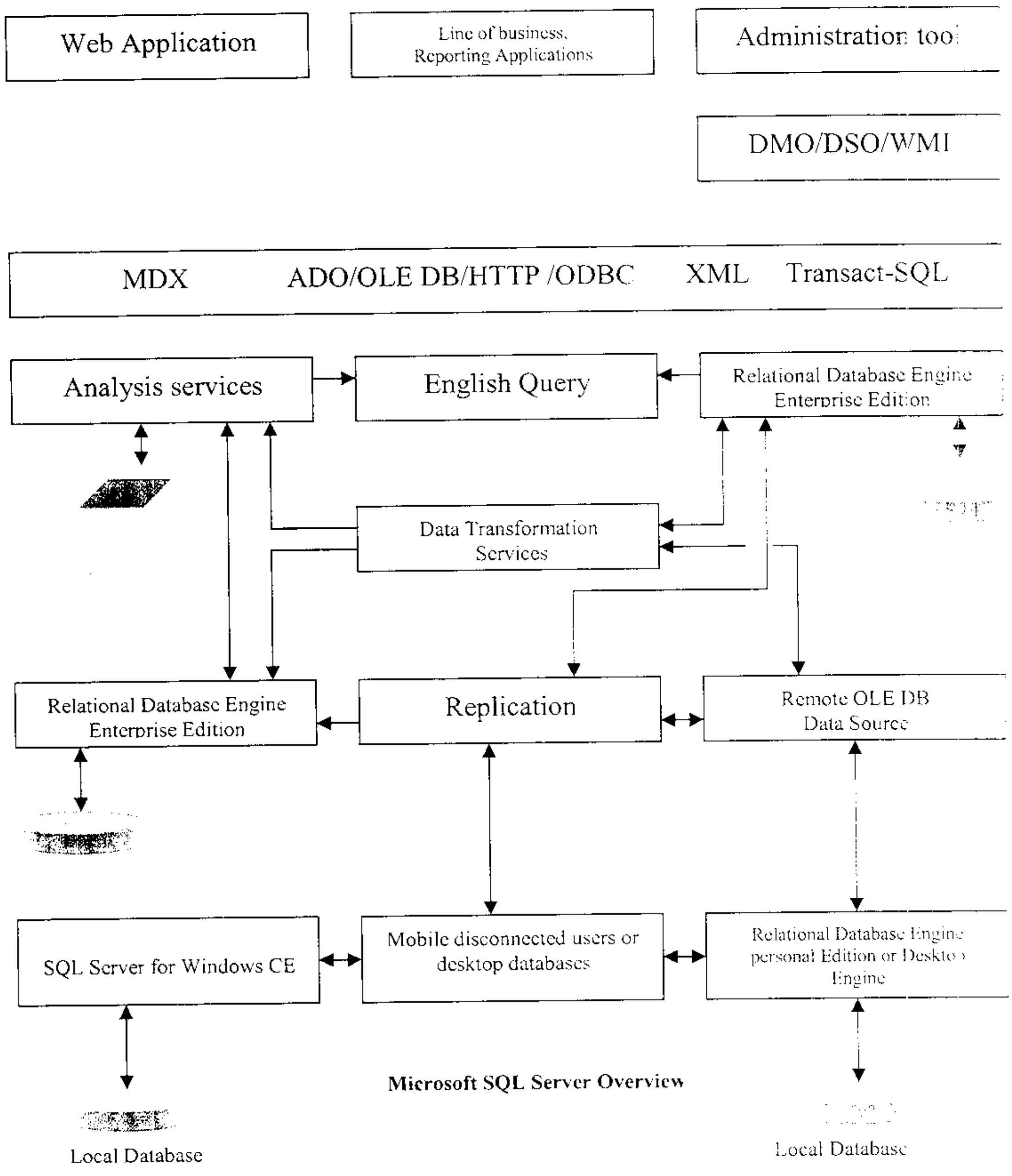
Decision Support Objects (DSO), a set of COM objects that encapsulates the administration functions for all of the entities in Analysis Services engine and multidimensional cubes.

Windows Management Instrumentation (WMI), SQL Server 2000 provides a SQL Server WMI provider that lets WMI applications get information on SQL Server databases and instances.

Additional Components

SQL Server 2000 provides several components that support important requirements of modern data storage systems. The data storage needs of today's large enterprises are very complex, and go beyond having a single OLTP system integrated with a single data warehouse or data mart. Increasing numbers of field personnel need to load sets of data, disconnect from the network, record their work autonomously during the day, then plug back in to the network and merge their records into the central data store at the end of the day. OLTP systems have to support the needs of both internal employees operating through an intranet and hundreds of thousands of customers placing orders through the

Web portal. Keeping data close to the workgroups or even individuals who primarily work on the data, and then replicating the data to a primary data store may minimize the overall processing load of the system.


Web Application
Line of business, Reporting Applications
Administration tool
DMO/DSO/WMI
MDX
ADO/OLE DB/HTTP /ODBC
XML
Transact-SQL
Analysis services
English Query
Relational Database Engine Enterprise Edition
Data Transformation Services
Relational Database Engine Enterprise Edition
Replication
Remote OLE DB Data Source
SQL Server for Windows CE
Mobile disconnected users or desktop databases
Relational Database Engine personal Edition or Desktop Engine
Local Database
Local Database

Microsoft SQL Server Overview

SQL Server 2000 replication allows sites to maintain multiple copies of data on different computers in order to improve overall system performance while at the same time making sure the different copies of data are kept synchronized. For example, a department could maintain the department sales data on a departmental server, but use replication to update the sales data in the corporate computer. Several mobile disconnected users can disconnect from the network, work throughout the day, and at the end of the day use merge replication to merge their work records back into the main database. These workers can be using SQL Server Personal Edition on notebook or laptop computers, or using SQL Server for Windows CE on Windows CE devices; all are supported by SQL Server replication. SQL Server replication also supports replicating data to data warehouses, and can replicate data to or from any data source that supports OLE DB access.

SQL Server 2000 Data Transformation Services (DTS) greatly improves the process of building OLAP data warehouses. Large OLTP databases are finely tuned to support the entry of thousands of business transactions at the same time. OLTP databases are also structured to record the details of every transaction. Trying to perform sophisticated analysis to discover trends in sales over a number of months and years would require scanning huge numbers of records, and the heavy processing load would drag down the performance of the OLTP databases. Data warehouses and data marts are built from the data in one or more OLTP systems that is extracted and transformed into something more useful for OLAP processing. OLTP detail rows are periodically pulled into a staging database, where they are summarized and the summary data is stored in a data warehouse or data mart. Data Transformation Services supports extracting data from one source of data, performing sometimes complex transformations of the data, and then storing the summarized, transformed data in another data source. The component greatly simplifies the process of extracting data from multiple OLTP systems and building it into an OLAP data warehouse or data mart.

DTS is not limited to being used to build data warehouses. It can be used any time one have to retrieve data from one data source, perform complex transformations on the data, and then store it in another data source. DTS is also not limited to working with SQL Server databases or Analysis Services cubes; DTS can work with any data source that can be accessed using OLE DB.

SQL Server 2000 English Query allows to build applications that can customize themselves to ad hoc user questions. An English Query administrator defines for the

English Query engine all of the logical relationships between the tables and columns of a database or the cubes in a data warehouse or data mart. An application can then present the user with a box where she can enter a character string with a question (written in English) about the data in the database or data warehouse. The application passes the string to the English Query engine, which analyzes the string against the relationships defined between the tables or cubes. English Query then returns to the application a SQL statement or MDX (multidimensional expression) query that will return the answer to the user's question.

Meta Data Services provides facilities for storing, viewing, and retrieving descriptions of the objects in the applications and system. Meta Data Services supports the MDC Open Information Model (OIM) specification defining a common format for storing descriptions of entities such as tables, views, cubes. or transformations, as well as the relationships between these entities. Application development tools that support OIM can use these descriptions to facilitate rapid development and interchange with other tools and applications. SQL Server components, such as Data Transformation Services packages and Analysis Services databases, can also be stored in the Meta Data Services repository

Using SQL Server 2000

An organization may use the SQL Server 2000 components to perform various tasks, for example:

Periodically, detailed OLTP data is extracted from the central databases by Data Transformation Services packages that scrub the data and build it into summary data that is then loaded into a data warehouse.

The senior managers and marketing personnel use Analysis Services to analyze the data warehouse for business trends that indicate possible opportunities that could be exploited or risks that must be minimized.

Relational Database Components

The database component of Microsoft® SQL Server™ 2000 is a Structured Query Language (SQL)-based, scalable, relational database with integrated Extensible Markup Language (XML) support for Internet applications. Each of the following terms describes a fundamental part of the architecture of the SQL Server 2000 database component:

Database

A database is similar to a data file in that it is a storage place for data. Like a data file, a database does not present information directly to a user; the user runs an application that accesses data from the database and presents it to the user in an understandable format.

Database systems are more powerful than data files in that data is more highly organized. In a well-designed database, there are no duplicate pieces of data that the user or application must update at the same time. Related pieces of data are grouped together in a single structure or record, and relationships can be defined between these structures and records.

When working with data files, an application must be coded to work with the specific structure of each data file. In contrast, a database contains a catalog that applications use to determine how data is organized. Generic database applications can use the catalog to present users with data from different databases dynamically, without being tied to a specific data format.

A database typically has two main parts: first, the files holding the physical database and second, the database management system (DBMS) software that applications use to access data. The DBMS is responsible for enforcing the database structure, including:

- Maintaining relationships between data in the database.
- Ensuring that data is stored correctly, and that the rules defining data relationships are not violated.
- Recovering all data to a point of known consistency in case of system failures.

Relational Database

Although there are different ways to organize data in a database, relational databases are one of the most effective. Relational database systems are an application of mathematical set theory to the problem of effectively organizing data. In a relational database, data is collected into tables (called relations in relational theory).

A table represents some class of objects that are important to an organization. Each table is built of columns and rows (called attributes and tuples in relational theory). Each column represents some attribute of the object represented by the table. Each row represents an instance of the object represented by the table. Relational database theory defines a process called normalization, which ensures that the set of tables define will organize the data effectively.

Scalable

SQL Server 2000 supports having a wide range of users access it at the same time. An instance of SQL Server 2000 includes the files that make up a set of databases and a copy of the DBMS software. Applications running on separate computers use a SQL Server 2000 communications component to transmit commands over a network to the SQL Server 2000 instance. When an application connects to an instance of SQL Server 2000, it can reference any of the databases in that instance that the user is authorized to access. The communication component also allows communication between an instance of SQL Server 2000 and an application running on the same computer. One can run multiple instances of SQL Server 2000 on a single computer.

SQL Server 2000 is designed to support the traffic of the largest Web sites or enterprise data processing systems. Instances of SQL Server 2000 running on large, multiprocessor servers are capable of supporting connections to thousands of users at the same time. The data in SQL Server tables can be partitioned across multiple servers, so that several multiprocessor computers can cooperate to support the database processing requirements of extremely large systems. These groups of database servers are called federations.

Although SQL Server 2000 is designed to work as the data storage engine for thousands of concurrent users who connect over a network, it is also capable of working as a stand-alone database directly on the same computer as an application. The scalability and ease-of-use features of SQL Server 2000 allow it to work efficiently on a single computer without consuming too many resources or requiring administrative work by the stand-alone user. The same features allow SQL Server 2000 to dynamically acquire the resources required to support thousands of users, while minimizing database administration and tuning. The SQL Server 2000 relational database engine dynamically tunes itself to acquire or free the appropriate computer resources required to support a varying load of users accessing an instance of SQL Server 2000 at any specific time. The SQL Server 2000 relational database engine has features to prevent the logical problems that occur if a user tries to read or modify data currently used by others.

Structured Query Language

To work with data in a database, one has to use a set of commands and statements (language) defined by the DBMS software. Several different languages can be used with relational databases; the most common is SQL. The dialect of SQL supported by Microsoft SQL Server is called Transact-SQL (T-SQL). T-SQL is the primary language used by Microsoft SQL Server applications.

Extensible Markup Language

XML is the emerging Internet standard for data. XML is a set of tags that can be used to define the structure of a hypertext document. XML documents can be easily processed by the Hypertext Markup Language, which is the most important language for displaying Web pages.

3.2.2 MICROSOFT SQL SERVER 2000 ANALYSIS SERVICE

Analysis Services

Microsoft® SQL Server™ 2000 extends and renames the former OLAP Services component, now called Analysis Services. Many new and improved features significantly enhance the analysis capabilities of the acclaimed OLAP Services introduced in SQL Server version 7.0. In this release, Analysis Services introduces data mining, which can be used to discover information in OLAP cubes and relational databases.

Introduction to OLAP

Cubes are the main objects in online analytic processing (OLAP), a technology that provides fast access to data in a data warehouse. A cube is a set of data that is usually constructed from a subset of a data warehouse and is organized and summarized into a multidimensional structure defined by a set of **dimensions** and **measures**.

Cubes : *A cube is a multidimensional structure that contains dimensions and measures. Dimensions define the structure of the cube, while measures provide the numerical values of interest to the end user. Cell positions in the cube are defined by the intersection of dimension members, and the measure values are aggregated to provide the values in the cells.*

Dimensions : *A structural attribute of a cube, which is an organized hierarchy of categories (levels) that describe data in the fact table. These categories typically describe a similar set of members upon which the user wants to base an analysis.*

Measures : *In a cube, a set of values that are based on a column in the cube's fact table and are usually numeric. Measures are the central values that are aggregated and analyzed.*

A cube provides an easy-to-use mechanism for querying data with quick and uniform response times. End users use client applications to connect to an Analysis server and query the cubes on the server. In most client applications, end users issue a query on a cube by manipulating the user interface controls, which determine the contents of the query. This spares end users from writing language-based queries. Precalculated summary data called **aggregations** provides the mechanism for rapid and uniform response times to queries. Aggregations are created for a cube before end users access it. The results of a query are retrieved from the aggregations, the cube's source data in the data warehouse, a copy of this data on the Analysis server, the client cache, or a combination of these sources. An Analysis server can support many different cubes, such as a cube for sales, a cube for inventory, a cube for customers, and so on.

Aggregation : *Aggregations are precalculated summaries of data that improve query response time by having the answers ready before the questions are asked.*

Every cube has a schema, which is the set of joined tables in the data warehouse from which the cube draws its source data. The central table in the schema is the fact table, the source of the cube's measures. The other tables are dimension tables, the sources of the cube's dimensions.

Cube Structure : *A cube's structure is defined by its measures and dimensions. They are derived from tables in the cube's data source. The set of tables from which a cube's measures and dimensions are derived is called the cube's schema. Every cube schema consists of a single fact table and one or more dimension tables. The cube's measures are derived from columns in the fact table. The cube's dimensions are derived from columns in the dimension tables.*

A cube is defined by the measures and dimensions that it contains. For example, a cube for sales analysis includes the measures Item_Sale_Price and Item_Cost and the dimensions Store_Location, Product_Line, and Fiscal_Year. This cube enables end users to separate Item_Sale_Price and Item_Cost into various categories by Store_Location, Product_Line, and Fiscal_Year.

Each cube dimension can contain a hierarchy of **levels** to specify the categorical breakdown available to end-users. For example, the Store_Location dimension includes the level hierarchy: Continent, Country, Region, State_Province, City, Store_Number. Each level in a dimension is of finer granularity than its parent. For example, continents contain countries, and states or provinces contain cities. Similarly, the hierarchy of the Fiscal_Year dimension includes the levels Year, Quarter, Month, and Day.

Level : *The name of a set of members in a dimension hierarchy such that all members of the set are at the same distance from the root of the hierarchy. For example, a time hierarchy may contain the levels Year, Month, and Day.*

Dimension levels are a powerful data modeling tool because they allow end users to ask questions at a high level and then expand a dimension hierarchy to reveal more detail. For example, an end user starts by asking to see Item_Cost values of products for the past three fiscal years. The end user may notice that 1998 Item_Cost values are higher than those in other years. Expanding the Fiscal_Year dimension to the Month level, the end user sees that Item_Cost values were especially high in the months January and August. The end user may then explore levels of the Store_Location dimension to see if a particular region contributed significantly to the high Item_Cost values, or may expand into the Product_Line dimension to see if Item_Cost values were high for a particular product group or product. This type of exploration, known as **drilldown**, is common in client applications.

Drill down/drill up : *A technique for navigating through levels of data ranging from the most summarized (up) to the most detailed (down). For example, when viewing the details of sales data by year, a user can drill down to display sales data by quarter, and further to display data by month.*

A cube can contain up to 128 dimensions, each with thousands or millions of members, and up to 1,024 measures. A cube with a modest number of dimensions and measures usually satisfies the requirements of end users.

There are several varieties of cubes in Microsoft® SQL Server™ 2000 Analysis Services. Although **regular cubes** possess the characteristics of cubes described in this topic and its subtopics, other varieties of cubes do not share all of these characteristics.

Regular cube : *A cube that is based on tables and has its own aggregations*

Cubes are immediately subordinate to the **database** in the object hierarchy. A database is a container for related cubes and the objects they share. You must create a database before you create a cube.

Database : *A database is a container for related cubes and the objects they share. These objects include data sources, shared dimensions, and database roles. If these objects are to be shared among multiple cubes, the objects and cubes must be within the same database.*

In the object hierarchy, the following objects are immediately subordinate to the cube:

Data sources

A cube has a single data source. It can be selected from the data sources in the database or created during cube creation. A cube's dimensions must have the same data source as the cube, but its partitions can have different data sources.

Measures

A cube's measures are not shared with other cubes. The measures are created when the cube is created. A cube can have up to 1,024 measures.

Dimensions

A cube's dimensions are either shared with other cubes in the database or private to the cube. Shared dimensions can be created before or during cube creation. Private dimensions are created when the cube is created. Although the term *cube* suggests three dimensions, a cube can have up to 128 dimensions.

Partitions

A single partition is automatically created for a cube when the cube is created. If you have installed Analysis Services for SQL Server 2000 Enterprise Edition, after creating a cube, you can create additional partitions in the cube.

Cube roles

Every cube must have at least one cube role in order to provide access to end-users. Cube roles are derived from database roles, which can be created before or after cube creation. Cube roles are created after cube creation.

After cubes are created, **partitions** or aggregations are usually the next objects to be created.

<u>Partitions</u> : *Partitions are used to store and manage precalculated aggregations and, sometimes, source data. They also offer flexibility in storing such data in multiple locations and in optimizing its access; they are also useful in managing the growth of cubes over time.*

Creating a cube involves three steps:

Definition

The definition of a cube is based on the analytical requirements of end users. To define a cube, select a fact table and identify measures within the fact table. Then select or create dimensions, each composed of one or more columns from another table. The Cube Wizard provides an easy way to define cubes. Cube Editor offers additional flexibility for defining and modifying cube structures.

Building Cubes

You can build a cube using the Cube Wizard or Cube Editor. The Cube Wizard takes you through the process in a series of steps. Cube Editor allows you to perform some of the steps in your own order. To build a cube using either the Cube Wizard or Cube Editor, you must specify the data source, fact table, measures, and dimensions for your cube.

The data source contains the fact table and dimension tables you want to include in your cube. A data source name identifies a database resource and parameters for its usage.

The fact table contains the measures you want to include in your cube. A fact table is the central table in a schema. It contains the numerical data (that is, measures) of main interest to end users of the cube. A fact table also contains foreign keys that are joined to primary keys in dimension tables.

The measures that you select are the ones that you want to make available to end-users. A measure contains numerical data (for example, Sales) viewed and analyzed by end users. Each measure corresponds to a column in the fact table. This column supplies the measure's values.

The dimensions that you select will also be made available to end-users. Dimensions are descriptive categories by which the measures can be separated for analysis. In tabular browsers, dimensions provide the column headings, row headings, and subheadings by which the measures are separated and displayed to end users. (In graphical browsers, they provide other types of descriptive labels but with the same function as in tabular browsers.) For example, the measure is Sales, and the dimensions are Time, Location, and Product. End users can separate Sales into various categories of Time, Location, and Product. Time provides headings for individual years and subheadings for months. Location and Product also supply a variety of headings and subheadings.

Each dimension is created from one or more columns in a dimension table. These columns supply the values of the dimension, and produce the column headings, row headings, and subheadings seen by end users.

Each dimension table contains a primary key that is joined to a foreign key in either the fact table or another dimension table.

You can also build virtual cubes, which combine elements of multiple, previously built cubes. When end users browse the virtual cube, they see the combined elements together as if they were in a single cube. One of the advantages of virtual cubes is that their definitions, but not their data, are stored. Thus, virtual cubes require much less storage space than regular cubes.

Aggregation design

After you define a new cube, you can **design its aggregations using the Storage Design** Wizard. Designing the aggregations specifies the summarization strategy.

Designing Storage Options and Aggregations : Use the Storage Design Wizard to quickly and easily set storage options and design aggregations for a partition. The wizard operates on a single partition at a time so that you can select different options and designs for each partition. If you start the wizard by selecting a multiple-partition cube, the wizard prompts you to select a partition. You can also start the wizard by selecting a single-partition cube or a partition. The wizard takes you through steps to specify storage and aggregation options for a partition.

Select a storage option if no aggregations exist or if you choose to replace existing aggregations. Each option is briefly described in the following table.

Storage option	**Description**
MOLAP	Multidimensional OLAP (MOLAP) stores aggregations and a copy of the partition's source data in a multidimensional structure on an Analysis server computer.
ROLAP	Relational OLAP (ROLAP) stores aggregations in a relational structure and leaves the partition's source data in its existing relational structure.
HOLAP	Hybrid OLAP (HOLAP) stores aggregations in a multidimensional structure on an Analysis server computer and leaves the partition's source data in its existing relational structure.

Each storage option has advantages and disadvantages. Aggregations are precalculated summaries of cube data that help enable Microsoft® SQL Server™ 2000 Analysis

Services to provide rapid query responses. Select a method of controlling the number of aggregations the wizard will design, and then let the wizard design the aggregations.

The goal is to design the optimal number of aggregations. This number should not only provide satisfactory response time, but also prevent excessive partition size. A greater number of aggregations produce faster response time but it also requires more storage space. Moreover, as the wizard designs more and more aggregations, earlier aggregations produce considerably larger performance gains than later aggregations. You can control the number of aggregations the wizard designs by one of the following methods available in the wizard:

Specify a storage space limit for the aggregations.

Specify a performance gain limit.

Stop the wizard manually when the displayed **Performance vs. Size** curve starts to level off at an acceptable performance gain.

The final step of the wizard allows you to process or defer processing. Processing creates the aggregations you design with the wizard, while deferring processing saves the designed aggregations for future processing, thus allowing design activities to continue without having to process. Depending on the size of the partition, processing may take considerable time.

Processing

After you design the aggregations of a new cube, process the cube with the **Full process** option. This action creates the aggregations.

After you create a cube, use Cube Editor to maintain it.

If, after you process a cube, you change it, or its source data changes, it is usually necessary to process the cube again. Different processing options are appropriate in different circumstances.

Processing Cubes : *When you process a cube, the aggregations designed for the cube are calculated and the cube is loaded with the calculated aggregations and data. Processing a cube involves reading the dimension tables to populate the levels with members from the actual data, reading the fact table, calculating specified aggregations, and storing the results in the cube. After a cube has been processed, users can query it.*

Cube Processing Options

Each of the following three processing options is appropriate in different circumstances:

- **Full Process**
- **Incremental update**
- **Refresh data**

In addition to these three mutually exclusive options, you can select a fourth option, **Incrementally update the dimensions of this cube**, in conjunction with any of these options. This option allows you to incrementally update the cube's dimensions as part of the cube processing.

Completely Processing a Cube

Full Process is the processing option used to perform a complete load of the cube. All dimension and fact table data is read and all specified aggregations are calculated. You must process a cube with the **Full Process** option when its structure is new or when the cube, its dimensions, or its measures have undergone structural changes. In addition, virtual and linked cubes also require complete processing after you build them, change their structure, or change one of their shared dimensions.

Potential exceptions are cubes in which only changing dimensions have been changed. A changing dimension does not always require its parent cubes to be processed with the **Full Process** option after the structure of the dimension is changed. However, processing with an alternative option may be required. Changing dimensions include virtual, parent-child, and relational OLAP (ROLAP) dimensions.

Processing a cube with the **Full Process** option can take a substantial amount of time if there is a large fact table and there are many dimensions with many levels and many items in each level. You do not need to load of dimension information if you use only preprocessed shared dimensions in cubes.

If there are changes in the data warehouse schema that affect the structure of cubes, you must change the structure of those cubes and then process them with the **Full Process** option. If there are changes in or additions to data in the data warehouse, you do not need to completely process cubes. Such changes can be incorporated into existing cubes using the **Incremental update** or **Refresh data** processing options, depending on how the data changed.

The **Full Process** option can be used while users continue to query a previously processed cube; however, after processing has completed, users need to disconnect and reconnect to reestablish access to the cube.

Incrementally Updating a Cube

An incremental update is appropriate when new data is to be added to a cube, but existing data has not changed and the cube structure remains the same. The **Incremental update** option adds new data and updates aggregations.

An incremental update does not affect the existing data that has already been processed. It usually requires significantly less time than processing with the **Full Process** option. An incremental update can be performed while users continue to query the cube; after the update is complete, users have access to the additional data without having to disconnect and reconnect.

Because an incremental update creates a temporary partition from the new data and merges it into an existing partition, it is necessary to understand the special considerations that apply to partitions before performing an incremental update.

Refreshing a Cube's Data

The **Refresh data** option causes a cube's data to be cleared and reloaded and its aggregations recalculated. This option is appropriate when the underlying data in the data warehouse has changed but the cube's structure remains the same.

The **Refresh data** option can be performed while users continue to query the cube; after the refresh has completed, users have access to the updated data without having to disconnect and reconnect.

Incrementally Updating Dimensions

The **Incrementally update the dimensions of this cube** option causes the cube's dimensions to be incrementally updated when the cube is processed. This option is valid with the **Full Process**, **Incremental update**, or **Refresh data** option. The **Incrementally update the dimensions of this cube** option is appropriate when rows have been added to any of the cube's dimension tables since the cube or dimension was last processed.

Cube Security

The type and scope of access to a cube by end users in a cube role is determined by the settings in the cube role. An end user can access only those cubes that are assigned a role containing that end user's user name.

A database role provides defaults for the cube roles of the same name, but some of these defaults can be overridden in the cube roles. After a database role is created, it can be assigned to any cube (including virtual and linked cubes) in the database. This action grants the end users in the database role access to the cube and creates a cube role with the same name as the database role. Database roles are assigned to cubes in the **Cubes** tab of the **Database Role** dialog box or in Cube Role Manager.

If a cube role does not specify restrictions on dimension members, end users in the cube role can view all members in the associated cube. If a dimension has been write-enabled, and the cube role has been granted read/write access to the dimension, the end users can also update members in the dimension. However, a database role or cube role can specify that some members can be viewed and updated and others cannot.

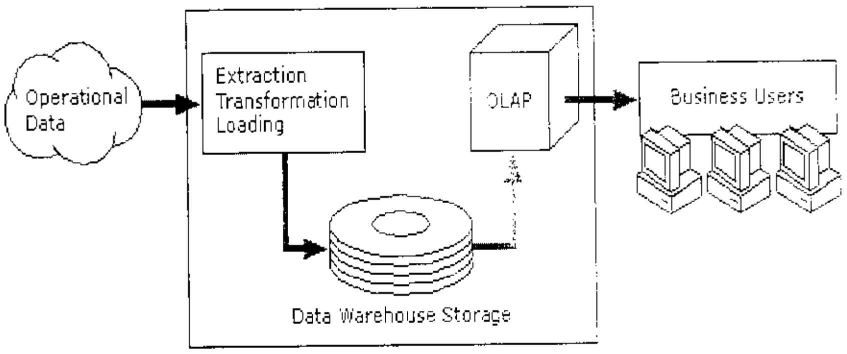
Similarly, by default, end users in a cube role can view all cells in the associated cube. If the cube has been write-enabled, and the cube role has been granted read/write access to the cube, the end users can also update cube cells. However, a cube role can specify that some cells can be viewed and updated and others cannot.

By default, end users in a cube role cannot drill through to any of the cube cells' source data. However, in a cube role you can grant this ability. If you grant this ability, you must enable drillthrough for the cube or for at least one of its partitions.

Data Warehousing and OLAP

Although sometimes used interchangeably, the terms *data warehousing* and *online analytical processing* (OLAP) apply to different components of systems often referred to as decision support systems or business intelligence systems. Components of these types of systems include databases and applications that provide the tools analysts need to support organizational decision-making.

A data warehouse is a database containing data that usually represents the business history of an organization. This historical data is used for analysis that supports business decisions at many levels, from strategic planning to performance evaluation of a discrete organizational unit. Data in a data warehouse is organized to support analysis rather than to process real-time transactions as in online transaction processing systems (OLTP).



OLAP technology enables data warehouses to be used effectively for online analysis, providing rapid responses to iterative complex analytical queries. OLAP's multidimensional data model and data aggregation techniques organize and summarize large amounts of data so it can be evaluated quickly using online analysis and graphical tools. The answer to a query into historical data often leads to subsequent queries as the analyst searches for answers or explores possibilities. OLAP systems provide the speed and flexibility to support the analyst in real time.

MDX

The Multidimensional Expressions (MDX) language is used to manipulate multidimensional information in Microsoft® SQL Server™ 2000 Analysis Services. MDX is defined in the OLAP extensions in OLE DB.

Similar to SQL in many respects, MDX provides a rich and powerful syntax for the retrieval and manipulation of multidimensional data, such as the data stored in cubes on the Analysis server. Analysis Services supports MDX functions in the definitions of calculated members, as well as a full language implementation for building local cubes and querying cube data using PivotTable® Service with OLE DB and Microsoft ActiveX® Data Objects (ADO).

Additionally, MDX supports the creation and registration of user-defined functions. One can create user-defined functions to operate on multidimensional data and accept arguments and return values in the MDX syntax.

The following topics provide more information about MDX.

MDX Overview

This section introduces Multidimensional Expressions (MDX) and explains some of the concepts behind its structure and syntax. It contains the following topics.

Introduction to MDX

MDX, an acronym for **Multidimensional** Expressions, is a syntax that supports the definition and manipulation of multidimensional objects and data. MDX is similar in many ways to the Structured Query Language (SQL) syntax, but is not an extension of the SQL language; in fact, some of the functionality that is supplied by MDX can be supplied, although not as efficiently or intuitively, by SQL.

As with an SQL query, each MDX query requires a data request (the SELECT clause), a starting point (the FROM clause), and a filter (the WHERE clause). These and other keywords provide the tools used to extract specific portions of data from a cube for analysis. MDX also supplies a robust set of functions for the manipulation of retrieved data, as well as the ability to extend MDX with user-defined functions.

MDX, like SQL, provides data definition language (DDL) syntax for managing data structures. There are MDX commands for creating (and deleting) cubes, dimensions, measures, and their subordinate objects.

Key Concepts in MDX

The purpose of Multidimensional Expressions (MDX) is to make accessing data from multiple dimensions easier and more intuitive.

Dimensions, Levels, Members, and Measures

Most languages used for data definition and manipulations, such as SQL, are designed to retrieve data in two dimensions: a column dimension and a row dimension.

Each table represents two-dimensional data. At the intersection of each row and column is a single element of data, called a field. The specific columns to be viewed in an SQL query are specified with a SELECT statement, and the rows to be retrieved are limited by a WHERE clause.

Multidimensional data, on the other hand, can be represented by structures with more than two dimensions. These structures, called cubes, have multiple dimensions. At the intersection of dimensions in a cube, there may be more than one element of data, called a measure. Each dimension is broken down into different levels, each of which is broken down further into members. For example, the Source dimension supplies the Eastern

Hemisphere level, which is broken down into four members, Africa, Asia, Australia, and Europe.

The querying of even simple data out of a multidimensional data source can be a complex task. A cube can have more than three dimensions, for example, or it may only have one dimension. The concepts of cubes, dimensions, levels, members, and measures are important to the understanding of MDX syntax.

Cells, Tuples, and Sets

As SQL returns a subset of two-dimensional data from tables, MDX returns a subset of multidimensional data from cubes.

The cube diagram illustrates that the intersection of multidimensional members creates cells from which one can obtain data. To identify and extract such data, whether it is a single cell or a block of cells, MDX uses a reference system called tuples. Tuples list dimensions and members to identify individual cells as well as larger sections of cells in the cube; because each cell is an intersection of all the dimensions of the cube, tuples can uniquely identify every cell in the cube. For the purposes of reference, measures in a cube are treated as a private dimension, named Measures, in the cube itself. For example, the following tuple identifies a cell in which the value is 400:

(Source.[Eastern Hemisphere].Africa, Time.[2nd half].[4th quarter], Route.Air, Measures.Packages)

The tuple uniquely identifies a section in the cube; it does not have to refer to a specific cell, nor does it have to encompass all of the dimensions in a cube. The following examples are all tuples of the cube diagram:

(Source.[Eastern Hemisphere])

(Time.[2nd half], Source.[Western Hemisphere])

These tuples provide sections of the cube, called slices, that encompass more than one cell.

An ordered collection of tuples is referred to as a set. In an MDX query, axis and slicer dimensions are composed of such sets of tuples. The following example is a description of a set of tuples in the cube in the diagram:

{ (Time.[1st half].[1st quarter]), Time.[2nd half].[3rd quarter]) }

In addition, it is possible to create a named set. A named set is a set with an alias, used to make the MDX query easier to understand and, if it is particularly complex, easier to process.

Axis and Slicer Dimensions

In SQL, it is usually necessary to restrict the amount of data returned from a query on a table. For example, one may want to see only two fields of a table with forty fields, and want to see them only if a third field meets a specific criteria. One can accomplish this by specifying columns in the SELECT statement, using a WHERE statement to restrict the rows that are returned based on specific criteria.

In MDX, those concepts also apply. A SELECT statement is used to select the dimensions and members to be returned, referred to as axis dimensions. The WHERE statement is used to restrict the returned data to specific dimension and member criteria, referred to as a slicer dimension. An axis dimension is expected to return data for multiple members, while a slicer dimension is expected to return data for a single member.

The terms "axis dimension" and "slicer dimension" are used to differentiate the dimensions of the cells in the source cube of the query, indicated in the FROM clause, from the dimensions of the cells in the result cube, which can be composed of multiple cube dimensions.

Comparison of SQL and MDX

The Multidimensional Expressions (MDX) syntax appears, at first glance, to be remarkably similar to the syntax of Structured Query Language (SQL). In many ways, the functionality supplied by MDX is also similar to that of SQL; with effort, one can even duplicate some of the functionality provided by MDX in SQL.

However, there are some striking differences between SQL and MDX, and one should be aware of these differences at a conceptual level. The following information is intended to provide a guide to these conceptual differences between SQL and MDX, from the point of view of an SQL developer.

The principal difference between SQL and MDX is the ability of MDX to reference multiple dimensions. Although it is possible to use SQL exclusively to query cubes in Microsoft® SQL Server™ 2000 Analysis Services, MDX provides commands that are designed specifically to retrieve data as multidimensional data structures with almost any number of dimensions.

SQL refers to only two dimensions, columns and rows, when processing queries. Because SQL was designed to handle only two-dimensional tabular data, the terms "column" and "row" have meaning in SQL syntax.

MDX, in comparison, can process one, two, three, or more dimensions in queries. Because multiple dimensions can be used in MDX, each dimension is referred to as an axis. The terms "column" and "row" in MDX are simply used as aliases for the first two axis dimensions in an MDX query; there are other dimensions that are also aliased, but the alias itself holds no real meaning to MDX. MDX supports such aliases for display purposes; many OLAP tools are incapable of displaying a result set with more than two dimensions.

In SQL, the SELECT clause is used to define the column layout for a query, while the WHERE clause is used to define the row layout. However, in MDX the SELECT clause can be used to define several axis dimensions, while the WHERE clause is used to restrict multidimensional data to a specific dimension or member.

In SQL, the WHERE clause is used to filter the data returned by a query. In MDX, the WHERE clause is used to provide a slice of the data returned by a query. While the two concepts are similar, they are not equivalent.

The SQL query uses the WHERE clause to contain an arbitrary list of items that should (or should not) be returned in the result set. While a long list of conditions in the filter can narrow the scope of the data that is retrieved, there is no requirement that the elements in the clause will produce a clear and concise subset of data.

In MDX, however, the concept of a slice means that each member in the WHERE clause identifies a distinct portion of data from a different dimension. Because of the organizational structure of multidimensional data, it is not possible to request a slice for multiple members of the same dimension. Because of this, the WHERE clause in MDX can provide a clear and concise subset of data.

The process of creating an SQL query is also different than that of creating an MDX query. The creator of an SQL query visualizes and defines the structure of a two-dimensional rowset and writes a query on one or more tables to populate it. In contrast, the creator of an MDX query usually visualizes and defines the structure of a multidimensional dataset and writes a query on a single cube to populate it. This could result in a multidimensional dataset with any number of dimensions; a one-dimensional dataset is possible, for example.

The visualization of an SQL result set is intuitive; the set is a two-dimensional grid of columns and rows. The visualization of an MDX result set is not as intuitive, however. Because a multidimensional result set can have more than three dimensions, it can be challenging to visualize the structure. To refer to such two-dimensional data in SQL, the name of a column and the unique identification of a row, in whatever method is appropriate for the data, are used to refer to a single cell of data, called a field. However, MDX uses a very specific and uniform syntax to refer to cells of data, whether the data forms a single cell or a group of cells.

Although SQL and MDX share similar syntax, the MDX syntax is remarkably robust, and it can be complex. However, because MDX was designed to provide a simple, effective way of querying multidimensional data, it addresses the conceptual differences between two-dimensional and multidimensional querying in a consistent and easily understood fashion.

Basic MDX

A multidimensional Expressions (MDX) command allows to query multidimensional objects, such as cubes, and return multidimensional datasets. This topic and its subtopics provide an overview of MDX queries.

As is the case with SQL, the author of an MDX query must determine the structure of the requested dataset before writing the query. The following topics describe MDX queries and the datasets they produce, and provide more detailed information about basic MDX syntax.

The Basic MDX Query

A basic Multidimensional Expressions (MDX) query is structured in a fashion similar to the following example:

```
SELECT [<axis_specification>

    [, <axis_specification>...]]

  FROM [<cube_specification>]

[WHERE [<slicer_specification>]]
```

Basic MDX Syntax - SELECT Statement

In MDX, the SELECT statement is used to specify a dataset containing a subset of multidimensional data. To discuss the various syntax elements of the MDX SELECT statement, this topic presents a basic MDX query example and breaks it down into its syntax elements, discussing the purpose and structure of each element.

To specify a dataset, an MDX query must contain information about:

The number of axes. One can specify up to 128 axes in an MDX query.

The members from each dimension to include on each axis of the MDX query.

The name of the cube that sets the context of the MDX query.

The members from a slicer dimension on which data is sliced for members from the axis dimensions.

This information can be complex. MDX syntax can provide such information in a simple and straightforward manner, using the MDX SELECT statement.

Basic MDX Query Example

The following MDX query example is used to discuss the various parts of basic MDX SELECT statement syntax:

```
SELECT
  { [Measures].[Unit Sales], [Measures].[Store Sales] } ON COLUMNS,
  { [Time].[1997], [Time].[1998] } ON ROWS
FROM Sales
WHERE ( [Store].[USA].[CA] )
```

The basic MDX SELECT statement contains a SELECT clause and a FROM clause, with an optional WHERE clause.

The SELECT clause determines the axis dimensions of an MDX SELECT statement. Two axis dimensions are defined in the MDX query example.

The FROM clause determines which multidimensional data source is to be used when extracting data to populate the result set of the MDX SELECT statement.

The WHERE clause optionally determines which dimension or member to use as a slicer dimension; this restricts the extracting of data to a specific dimension or member. The MDX query example uses a WHERE clause to restrict the data extract for the axis dimensions to a specific member of the Store dimension.

The MDX SELECT statement supports other optional syntax, such as the WITH keyword, and the use of MDX functions to construct members by calculation for inclusion in an axis or slicer dimension

The syntax format of the MDX SELECT statement is similar to that of SQL syntax; however, there are several obvious differences:

MDX syntax distinguishes sets by surrounding tuples or members with braces (the { and } characters

MDX queries can have up to 128 axis dimensions in the SELECT statement, but only the first 5 axes have aliases. An axis can be referred to by its ordinal position within an MDX query or by its alias, if it has an alias assigned to it. In the MDX query example, the COLUMNS and ROWS axis aliases are used. The MDX query could also have been written in the following fashion, using the ordinal position of each axis:

```
SELECT

  { [Measures].[Unit Sales], [Measures].[Store Sales] } ON AXIS(0),

  { [Time].[1997], [Time].[1998] } ON AXIS(1)

FROM Sales

WHERE ( [Store].[USA].[CA] )
```

As with an SQL query, the FROM clause names the source of the data for the MDX query. However, unlike an SQL query, the FROM clause in an MDX query is restricted to a single cube. Information from other cubes can be retrieved, however, on a value-by-value basis using the **LookupCube** function.

The WHERE clause is used to describe the slicer dimensions. If a dimension is not mentioned as part of the WHERE clause, Microsoft® SQL Server™ 2000 Analysis Services assumes that any dimension not assigned to an axis dimension is a slicer dimension, and the dimension is filtered on its default members. The WHERE clause can change the filtering process for specified dimensions, allowing fine control of included data.

Members, Tuples, and Sets

Before proceeding on the creation of a Multidimensional Expressions (MDX) query, one should understand the definitions of members, tuples and sets, as well as the MDX syntax used to construct and refer to these elements.

Members

A member is an item in a dimension representing one or more occurrences of data. Think of a member in a dimension as one or more records in the underlying database whose value in this column falls under this category. A member is the lowest level of reference when describing cell data in a cube.

For example, the Time.[2nd half].[3rd quarter] member.

The bracket characters, [and], are used if the name of a member has a space or a number in it. Although the Time dimension is one word, bracket characters can also be used around it as well; the member could also be represented as:

```
[Time].[2nd half].[4th quarter]
```

Tuples

A tuple is used to define a slice of data from a cube; it is composed of an ordered collection of one member from one or more dimensions. A tuple is used to identify specific sections of multidimensional data from a cube; a tuple composed of one member from each dimension in a cube completely describes a cell value. Put another way, a tuple is a vector of members; think of a tuple as one or more records in the underlying database whose value in these columns falls under these categories.

In MDX, tuples are syntactically constructed depending upon their complexity. If a tuple is composed of only one member from a single dimension, often referred to as a simple tuple, the following syntax is acceptable.

```
Time.[2nd half]
```

If a tuple is composed of members from more than one dimension, the members represented by the tuple must be enclosed in parentheses, as demonstrated in the following example.

```
(Time.[2nd half], Route.nonground.air)
```

A tuple composed of a single member can also be enclosed in parentheses, but this is not required. Tuples are often grouped together in sets for use in MDX queries.

Sets

A set is an ordered collection of zero, one or more tuples. A set is most commonly used to define axis and slicer dimensions in an MDX query, and as such may have only a single tuple or may be, in certain cases, empty. The following example shows a set of two tuples:

```
{ (Time.[1st half], Route.nonground.air), (Time.[2nd half], Route.nonground.sea) }
```

A set can contain more than one occurrence of the same tuple. The following set is acceptable:

```
{ Time.[2nd half], Time.[2nd half] }
```

A set refers to either a set of member combinations, represented as tuples, or to the values in the cells that the tuples in the set represent, depending on the context of usage for the set. In MDX syntax, tuples are enclosed in braces to construct a set.

Axis and Slicer Dimensions

When formulating a Multidimensional Expressions (MDX) query, an application typically looks at the cubes and divides the set of dimensions into two subsets:

Axis dimensions, for which data is retrieved for multiple members.

Slicer dimensions, for which data is retrieved for a single member.

Because axis and slicer dimensions can be constructed from multiple dimensions of the cube to be queried, these terms are used to differentiate the dimensions employed by the cube to be queried from the dimensions created in the cube returned by an MDX query.

For example, assume that a cube exists, named TestCube, with two simple dimensions named Route and Time. Because the measures of the cube are part of the Measures dimension, this cube has three dimensions in all. The query is to provide a matrix in which the Packages measure can be compared across routes and times.

In the following MDX query example, the Route and Time dimensions are used as axis dimensions and the Measures dimension is used as the slicer dimension. The Members function indicates that the members of the dimension or level are to be used to construct a set, instead of having to explicitly state each member of a given dimension or level in an MDX query.

```
SELECT
  { Route.nonground.Members } ON COLUMNS,
  { Time.[1st half].Members } ON ROWS
FROM TestCube
WHERE ( [Measures].[Packages] )
```

The resulting grid of values would resemble the following table, showing the value of the Packages measure at each intersection of the COLUMNS and ROWS axis dimensions.

	air	sea
1st quarter	60	50
2nd quarter	45	45

MDX evaluates the axis and slicer dimensions first, building the structure of the result cube before retrieving the information from the cube to be queried.

Programming Analysis Services Applications

Microsoft® SQL Server™ 2000 Analysis Services provides support to create and integrate custom applications that enhance the online analytical processing (OLAP) and data mining installation.

Analysis Services includes the Analysis server and PivotTable® Service. The Analysis server manages and stores multidimensional information and serves client application requests for OLAP data. PivotTable Service is an OLE DB for OLAP provider that connects client applications to the Analysis server and manages offline cubes. A repository of meta data contains definitions of OLAP data objects such as cubes and their elements.

An object model, Decision Support Objects (DSO), provides support for the Analysis Manager user interface and for custom applications that manage OLAP meta data and control the server. An interface, **IOlapAddIn**, enables the applications to extend and interact with the user interface. PivotTable Service provides access to OLAP data from the server and the ability to create local cubes.

One can create applications that:

Manage the Analysis server and create and maintain OLAP and data mining objects such as cubes, dimensions, security roles, and data mining models.

Extend the user interface by adding new objects to the object tree pane and by adding and responding to new menu choices.

Connect to the Analysis server, query data in cubes, and create local cubes.

Combine any or all of these functions.

Analysis Services Architecture

Microsoft® SQL Server™ 2000 Analysis Services includes the Analysis server and PivotTable® Service. The Analysis server creates and manages multidimensional data cubes for online analytical processing (OLAP) and provides multidimensional data to PivotTable Service, which in turn provides this data to clients through Microsoft ActiveX® Data Objects (Multidimensional) (ADO MD) and OLE DB for OLAP provider services.

The server stores cube meta data (cube definition specifications) in a repository. Completed cubes can be stored in a variety of storage modes: as multidimensional database files (MOLAP), as tables in a relational database (ROLAP), or as a hybrid of multidimensional database files and relational tables (HOLAP).

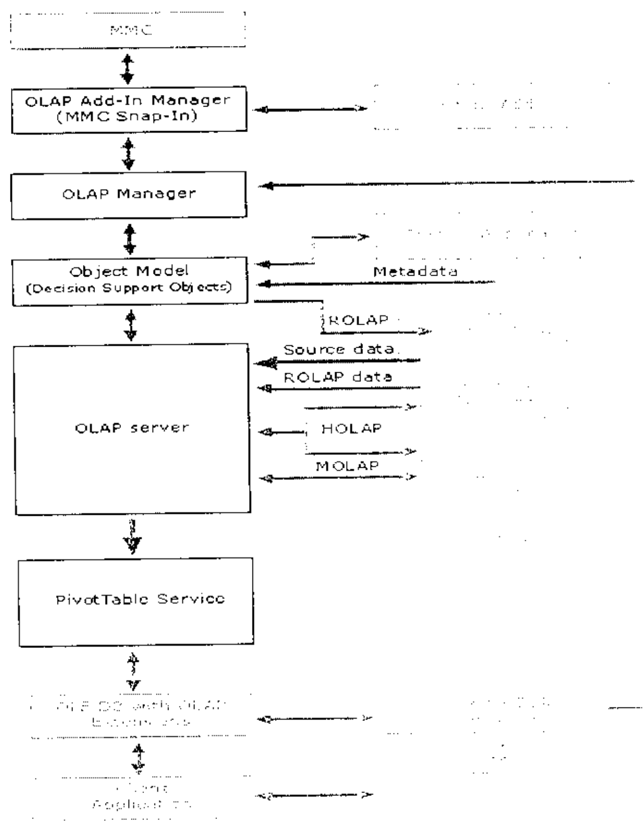
Source data for multidimensional cubes resides in relational databases where the data has been transformed into a star or snowflake schema typically used in OLAP data warehouse systems. Analysis Services can work with many relational databases that support connections using ODBC or OLE DB. When used as part of SQL Server 2000, Analysis Services offers enhanced security and other capabilities. The Data Transformation Services (DTS) feature of SQL Server 2000 provides a means to manage the data warehouse from which Analysis Services creates cubes.

Control of the server is accomplished through the Analysis Manager user interface, or through custom applications developed using the Decision Support Objects (DSO) object model. DSO controls the creation and management of cubes by the server, and manages the cube meta data in the repository. The object model is used by the Analysis Manager program that provides the user interface through a snap-in to Microsoft Management Console (MMC). The DSO object model can be used by applications written in Microsoft

Visual Basic® to provide custom programmatic control of the server. One can also develop custom applications to interact with the Analysis Manager user interface.

The following diagram illustrates the elements and functions of the Analysis server and its use of PivotTable Service to provide multidimensional data to client consumer applications. The Analysis Manager user interface uses PivotTable Service to obtain multidimensional data from the server for browsing by the server administrator.

OLAP Services System Architecture (with PivotTable Service)



3.2.3 Active Server Page

Active Server Pages (ASP) is a server-side scripting environment that you can use to create dynamic Web pages or build powerful Web applications. ASP pages are files that contain HTML tags, text, and script commands. ASP pages can call ActiveX components to perform tasks, such as connecting to a database or performing a business calculation.

With ASP, you can add interactive content to your Web pages or build entire Web applications that use HTML pages as the interface to your customer. ASP applications are easy to develop and modify.

The topics introduces Active Server Pages, explain the basic concepts of scripting with Active Server Pages, and discuss more complex application issues such as how to maintain state.

It offers the following

1. Browser independence. ASP can run complex page building login on the server and send only results to the client.

2. Database constructed pages that allow viewing, updating and additions to server databases

3. Easy to user components running on the server (not the client). Build in third party components that require no browser scripting ability. Yet accomplish complex tasks that are difficult with browser scripting.

4. Display different web pages, depending on the capability of a user's browser

5. Increased performance - The applications run on powerful server forms which allow for faster, more efficient computing.

6. Remote accessibility - You can access the applications from any location in the world with simply an internet connection and a standard web browser.

7. Save money on hardware - Because the applications are on run on server forms, your hardware needs are unimportant. There is no need to be concerned about RAM, disk space, or increasing CPU speed.

4.1 INTRODUCTION

OLAP systems are created for the purpose of analyzing an organizations data. They import data from the OLTP systems and other sources, including sources from outside of the organization. The importing is done through Data Transformation Service (DTS). An OLAP database is a structure that is used to store a set of related cubes. The cubes store data in a format that optimizes analytical queries. These queries often summarize information from many records and many tables. The goal of an OLAP system is to provide a browsing tool that allows an analyst to view the data from al. useful perspective.

The Internet and corporate Intranets provide excellent network infrastructures on which to base an OLAP application. The application built is not overly complex because it is used only as a Prototype Model. The application is based on Microsoft's IIS Server as the Web Server. The application makes use of ADO and ADOMD connection objects. Microsoft OLAP Services, Active Server Pages, and HTML.

Every section of the System Design and Development is divided to two main sections. The first section presents the development of OLAP cube. The second section presents the development of OLAP Web Client.

The Architecture for the Web Client:

The development guidelines and some design requirements to be established are given below. Two important design objectives and assumptions are.

The client will be as "thin" as possible. ActiveX Controls, Java applets, DHTML and XML will not be used.

The server will make maximal use of Active Server Pages (ASP), and not use Web classes or custom server-side programming components.

To begin the development of OLAP Web client for a specific situation with well-defined user requirements, use of more current Web Programming techniques would be entirely appropriate.

The second critical assumption is the decision to limit the server-side programming to ASP. Deciding to use Microsoft's IIS Web Server can be justified by the fact that the OLAP architecture being discussed here is almost entirely Microsoft-based.

4.2 INPUT DESIGN

The fundamental design goal for OLAP RIMS Express is to be the de-facto equity research and portfolio analysis repository. The application gathers information from multiple sources, both internal and external, then stores and categorizes it. With this information, different analytical views can be presented appropriate for the Analyst, regardless of where it is stored. OLAP RIMS Express has made it easier for users to access a much greater variety of data in a more timely fashion. This spreadsheet viewing of data at the desktop level dramatically empowers the analysts and Portfolio Managers. As financial data becomes increasingly abundant, the underlying system architecture for the application must be robust enough to handle not just what is available today, but rather factor in future information growth along with the additional data media options such as voice and video. OLAP RIMS Express is designed with the intent of accommodating this future growth and by accommodating itself into the intermediate layers.

Query page

1. What is it?

The Query page is one of the most flexible and powerful page in OLAP RIMS Express. Users can view a large portion of OLAP RIMS Express data through the Query page. The user can selects a list of dimensions and a set of measures to view. Currently there are 105 portfolios, 188 CM Sectors, 242 Regions, and flexible time from the year 2002 to choose from. The analysis can be done by specifying filtering conditions.

2. What data is available?

The data for the Query page comes from internal data source such as RCMdb and external data vendors, including IBES, Factset, Bloomberg, and Muller can be added. Majority of the data items exposed on the RIMS Express site are also available for Querying; including earnings, valuation, holdings, price, votes and reference data, in total, there are around 4.85 Crore records that are currently available for access through this Query page.

3. How it is being used?

The Query page is often used as a fast place to retrieve data from the cube. One can select any hierarchy level from the dimension. Additionally, for analysis one can select certain measures so that it would be highlighted.

4. Going Forward

The OLAP RIMS Express is extremely flexible, new feature and new dimensions can be added. In the future, users will be able to use OLAP functions. The functionality could also be expanded to the portfolio level, so that users could view lists of portfolios or benchmarks and relevant aggregate data.

The Result page can be improved with the help of thick client architecture. The Performance can be improved if appropriate hardware configuration is used. As with any modern GUI based application, the user can easily sort the data in their report by clicking column headers, such features can be added. Additionally, buttons can be provided to print the data or to download to Excel. Users could save reports, delete reports, or modify existing reports. The saved reports could be analyzed. As an additional value-added process, saved report can be scheduled to be printed automatically on a pre-set interval or if desired, content of report can be output to network drive as either text file or Excel format.

4.3 DATABASE DESIGN

1. Table Name : **country_region**
Purpose : Stores the details of the country.

SL. No.	Column Name	Data Type (length)	Description
1.	iso_country_id	char (2)	ISO Country ID, comes from MSCI, RCMdb, IBES in that order. Note IBES's country_id needs to be mapped in order to get iso_country_id, ex iso_country_id US has IBES country_id NA
2.	country_nm	Varchar (50)	The name of the country corresponding to the iso_country_id.
3.	msci_rgn_cd	char (2)	MSCI region code comes from MSCI table. It is the region code mapped against country ids.
4.	msci_rgn_nm	char (25)	The Region Name corresponding to the msci_rgn_cd.
5.	msci_rgn_nm_long	varchar (28)	Region Name (long); Computed as msci_rgn_cd '-' msci_rgn_nm from same table
6.	Currency	char (3)	Currency of the country.
7.	reg_code	char (4)	The region code from msci_region_test is updated into this column.

2. Table Name : **olap_pfolio_names**
Purpose : Stores the details of the portfolio dimension.

SL. No.	Column Name	Data Type (length)	Description
1.	pfolio_id	char (2)	Primary key, from RCMdb for portfolios, rims creates for other types (indexes, aggregates, models,...)
2.	rims_id	Int	RIMS security
3.	mstr_shrt_nm_id	varchar(16)	Short name of the portfolio.
4.	name	varchar(50)	The stocks name relating to the portfolio.

3. Table Name : **olap_time_table**
Purpose : Dimension table that stores time factor.

SL. No.	Column Name	Data type (length)	Description
1.	time_id	Int(4)	Primary key, from olapdb for time dimension
2.	monthno	Int(4)	Month denoting from year 2002
3.	quarterno	Int(4)	Quarter denoting every 3 months from year 2002
4.	yearno	Int(4)	Year starting from 2002.

4. Table Name : **cm_ind**
Purpose : This table contains CM sector information.

SL. No.	Column Name	Data type (length)	Description
1.	subind_id_hs	char (3)	House subindustry classification, equals concatenation of RCMdb's mjr_ind_id and mnr_ind_id, maintained in RIMS admin section, join to house industries table to get mjr_ind, broad ind ids and names
2.	mnr_ind_id	char (1)	Updated from RCMdb's table mnr_ind that represents a minor industry classification.
3.	mjr_ind_id	char (2)	Updated from RCMdb's table mjr_ind that represents a major industry classification.
4.	broad_ind_id	char (2)	Updated from RCMdb's table broad_ind that represents a broad industry classification.
5.	mnr_ind_nm_long	varchar (44)	Minor Industry long Name. Computed from mjr_ind_id + mnr_ind_id + '-' + mnr_ind_nm
6.	mjr_ind_nm_long	varchar (43)	Major Industry long Name Computed from mjr_ind_id + '-' + mjr_ind_nm
7.	broad_ind_nm_long	varchar (33)	Broad Industry long Name Computed from convert(varchar(2),convert(int,broad_ind_id)) + '-' + broad_ind_nm
8.	mnr_ind_nm	varchar (40)	Minor Industry name
9.	Mjr_ind_nm	varchar (40)	Major Industry Name
10.	broad_ind_nm	varchar (30)	Broad industry Name

5. **Table Name** : **ibes_ind**
   Purpose : This table provides the information about the Industry with sector details.

Sl. no.	Column Name	Type (length)	Description
1	ibes_ind_id	Char(4)	IBES industry id from IBES, see ibes_tkr_general info, ibes_group, and ibes_industry_names.si_code field, for industry name and sector and industry id and name join to ibes industries table (based on IBES..ibes_industry_names table)
2	sect_no	tinyint	Sector Number/id as updated from the table ibes..ibes_industry_names. sect_no
3	ind_no	Tinyint	Industry Number/id as updated from the table ibes..ibes_industry_names. ind_no
4	sect_abbr	Char(8)	Sector abbreviation as updated from table ibes..ibes_industry_names. sect_abbr
5	ind_abbr	Char(8)	Industry abbreviation as updated from table ibes..ibes_industry_names. ind_abbr
6	sect_name	Char(24)	Sector Name as updated from table ibes..ibes_industry_names. sect_name
7	ind_name	Char(24)	Industry Name as updated from table ibes..ibes_industry_names. ind_name
8	sect_name_long	Varchar(55)	Sect_no+ '-' + sect_name
9	ind_name_long	Varchar(55)	ind_no+ '-' + ind_name

6. **Table Name** : **gic_ind**
   Purpose : Stores the details of industry grouping of the stock like industry number, name, sector number and name.

Sl. no.	Column Name	Type (length)	Description
1	gic_ind_id	char(4)	GIC industry classification, from MSCI, cm map and exceptions see stored procedure update_sec_gic_ind. Join to GIC industries table to get sector number and industry and sector names. Updated from RIMS tables gic_ind and gic_sect tables.
2	gic_sect_id	char(1)	The GIC sector id corresponding to the gic_ind_id. Updated from RIMS tables gic_ind and gic_sect tables.
3	gic_ind_nm_long	varchar(55)	The GIC industry long name computed by gic_ind_id + gic_ind_nm. Updated from RIMS tables gic_ind and gic_sect tables.
4	gic_sect_nm_long	varchar(52)	The GIC sector long name. Computed as [gic_ind_id] + '-' + [gic_ind_nm]
5	gic_ind_nm	varchar(50)	The GIC industry number. [gic_sect_id] + '-' + [gic_sect_nm]
6	gic_sect_nm	varchar(50)	The GIC sector name corresponding to the gic_sect_id. Updated from RIMS tables gic_ind and gic_sect tables.

7. **Table Name** : **msci_ind**
Purpose : This table provides the details of the Sub Industries.

[illegible]	[illegible]	[illegible]	[illegible]
1	sub_indus try_code	Char(8)	The Subindistry id that is updated from table msci..msci_sub_industry. sub_industry_code
2	industry_ group_co de	Char(4)	Indicates to which Industry this subindustry belongs updated from table msci..msci_sub_industry. industry_group_code
3	sector_co de	Char(2)	Indicates to which Sector this subindustry belongs updated from table msci..msci_sub_industry. sector_code
4	sub_indus try_name _long	Varchar(137)	Long Name of this Sub Industry Computed as sub_industry_code + '-' + sub_industry_name
5	industry_ group_na me_long	Varchar(133)	Long Name of the Industry to which this Subindustry belongs Computed as industry_group_code+ '-' + industry_group_name
6	sector_na me_long	Varchar(131)	Long Name of the Sector to which this Subindustry belongs Computed as sector_code+'-'+ sector_name
7	sub_indus try_name	Varchar(128)	Name of the Sub Industry as updated from table msci..msci_sub_industry. sub_industry_name
8	industry_ group_na me	Varchar(128)	Name of the Industry to which this Subindustry belongs updated msci..msci_industry_group. industry_group_name from table
9	sector_na me	Varchar(128)	Name of the Sector to which this Subindustry belongs as updated from table msci..msci_sector. sector_name

8. **Table Name** : **ftse_ind**
Purpose : This table contains FTSE industry details

SL. No.	**Column Name**	**Data Type (length)**	**Description**
1.	ftse_subind_i d	Char (3)	FTSE subindustry classification, from London, join to FTSE industries table to get sub industry and sector ids and names
2.	ftse_ind_id	Char (2)	FTSE industry classification, from London, join to FTSE industries table to get industry and sector ids and names
3.	ftse_sect_id	Char (2)	FTSE Sector ID
4.	ftse_subind_n m_long	Varchar (84)	FTSE Sub Industry ID Computed as ftse_subind_id + '-' + ftse_subind_nm
5.	ftse_ind_nm_l ong	Varchar (83)	FTSE industry long name Computed as ftse_ind_id + '-' + ftse_ind_nm

6.	ftse_sect_nm_long	Varchar (83)	FTSE sector long name Computed as ftse_sect_id + '-' + ftse_sect_nm
7.	ftse_subind_nm	Varchar (80)	FTSE sub industry name corresponding to the ftse_subind_id
8.	ftse_ind_nm	Varchar (80)	FTSE industry name corresponding to the ftse_ind_id
9.	ftse_sect_nm	Varchar (80)	FTSE sector name corresponding to the ftse_sect_id

9. Table Name **: olap_fact_table**
Purpose : This table contains keys and measures and becomes an OLAP fact table

SL. No.	Column Name	Data Type (Length)	Description
1.	pfolio_id	Int(4)	Foreign key, from olapdb to olap_pfolio_names
2.	time_id	Int(4)	Foreign key, from olapdb to olap_time_table
3.	history_dt	Datetime	Denotes the last update
4.	iso_country_id	Char(2)	Foreign key, from olapdb to country_region
5.	hs_subind_id	Char(3)	Foreign key, from olapdb to cm_ind
6.	ftse_subind_id	Char(3)	Foreign key, from olapdb to ftse_ind
7.	ibes_ind_id	Char(4)	Foreign key, from olapdb to ibes_ind
8.	msci_sp_subind_id	Char(8)	Foreign key, from olapdb to msci_ind
9.	gic_ind_id	Char(4)	Foreign key, from olapdb to gic_ind
10.	mkt_cap_usd_mil	Float(8)	Market capital in US Dollars in Millions
11.	shares_held_cm	Float(8)	Shares held by CM for a Stock
12.	price	Float(8)	latest closing price in local currency. updated by price feeds several times per day
13.	price_1d	Float(8)	price in local currency from previous day
14.	mtd_return	Float(8)	Percentage change between the current price and month end price.
15.	qtd_return	Float(8)	Percentage change between the current price and quarter end price.
16.	ytd_return	Float(8)	Percentage change between the current price and Year end price.
17.	twelve_mtd_return	Float(8)	Percentage change between the current price and 12 months end price.
18.	eps_q_curr	Float(8)	Computed as : case (datepart(quarter,getdate())) when 1 then [eps_y0_q1] when 2 then [eps_y0_q2] when 3 then [eps_y0_q3] when 4 then [eps_y0_q4] end
19.	mkt_pe_curr1	Float(8)	P/E of relative market for current year. updated from rims..indx table based on earnings period (midpoint or FY) and chosen market
20.	anlst_vote	Float(8)	Analyst vote, updated from Stocks/Votes page or Research page

4.4 PROCESS DESIGN

The Architecture for the Web Client:

The development guidelines and some design requirements to be established are given below. Two important design objectives and assumptions are.

The client will be as "thin" as possible. ActiveX Controls, Java applets, DHTML and XML will not be used.

The server will make maximal use of Active Server Pages (ASP), and not use Web classes or custom server-side programming components.

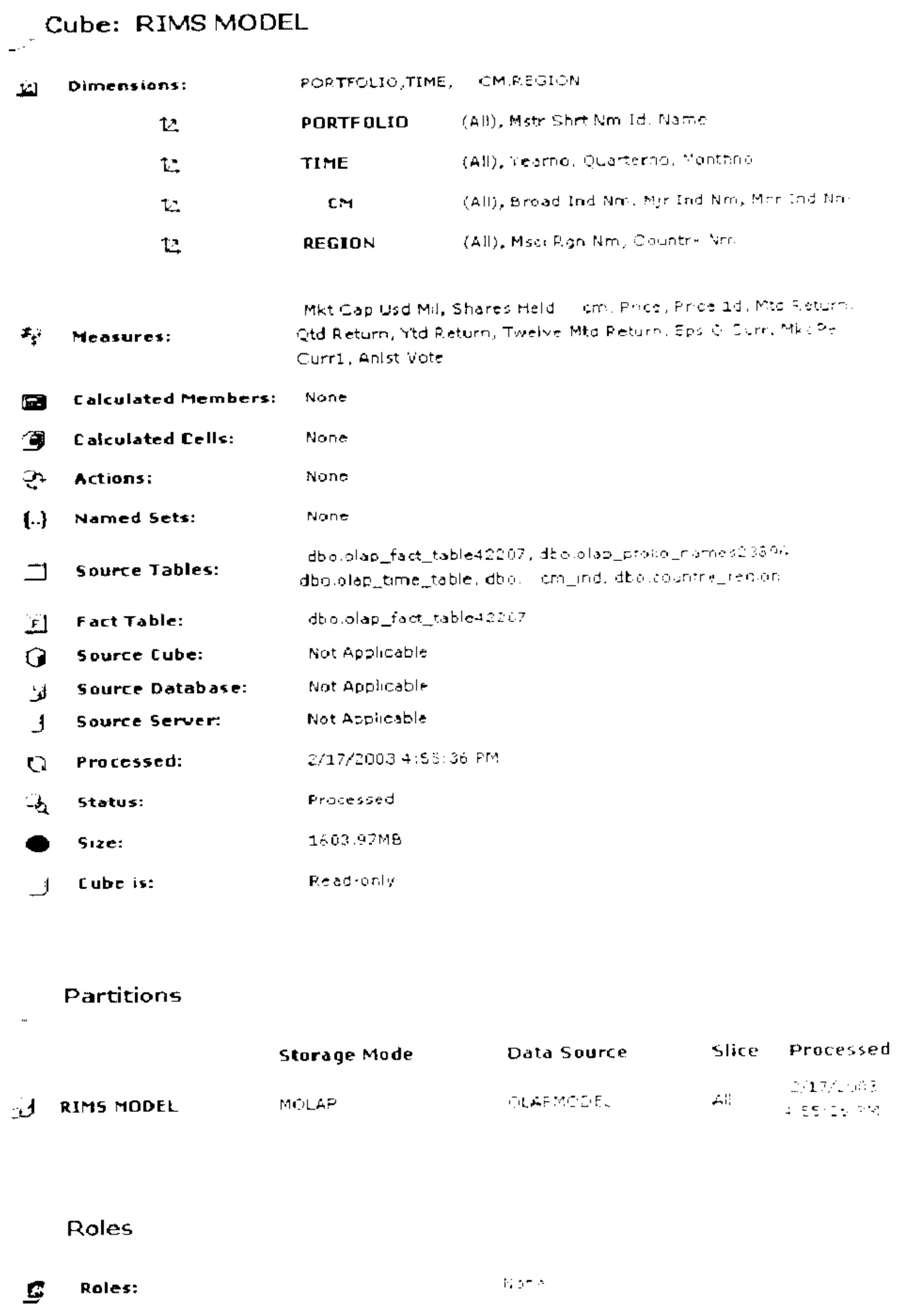
To begin the development of OLAP Web client for a specific situation with well-defined user requirements, use of more current Web Programming techniques would be entirely appropriate.

The second critical assumption is the decision to limit the server-side programming to ASP. Deciding to use Microsoft's IIS Web Server can be justified by the fact that the OLAP architecture being discussed here is almost entirely Microsoft-based.

The functional requirements of OLAP Web Client are very simple.

- The application will Connect with a server, database, and cube and display cube metadata to the user on a Web page.
- Use html form components and variables to allow the user to specify the parameters of an MDX query. The user will not be required to directly create or edit the MDX query.
- Provide limited capabilities for MDX functions.
- Execute the MDX query and display the results to the user in an html table.

Meta Data



Cube: RIMS MODEL

Dimensions:	PORTFOLIO,TIME, CM,REGION	
	PORTFOLIO	(All), Mstr Shrt Nm Id, Name
	TIME	(All), Yearno, Quarterno, Monthno
	CM	(All), Broad Ind Nm, Mjr Ind Nm, Mnr Ind Nm
	REGION	(All), Msci Rgn Nm, Country Nm
Measures:	Mkt Cap Usd Mil, Shares Held cm. Price, Price 1d, Mtd Return, Qtd Return, Ytd Return, Twelve Mtd Return, Eps Q Curr, Mkt Pe Curr1, Anlst Vote	
Calculated Members:	None	
Calculated Cells:	None	
Actions:	None	
Named Sets:	None	
Source Tables:	dbo.olap_fact_table42207, dbo.olap_protio_names23896 dbo.olap_time_table, dbo. cm_ind, dbo.country_region	
Fact Table:	dbo.olap_fact_table42207	
Source Cube:	Not Applicable	
Source Database:	Not Applicable	
Source Server:	Not Applicable	
Processed:	2/17/2003 4:55:36 PM	
Status:	Processed	
Size:	1603.97MB	
Cube is:	Read-only	

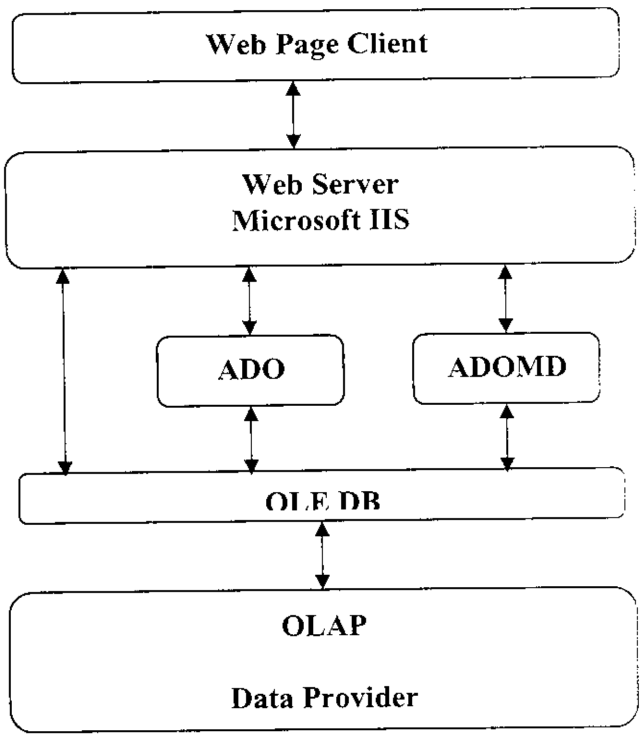
Partitions

	Storage Mode	Data Source	Slice	Processed
RIMS MODEL	MOLAP	OLAPMODEL	All	2/17/2003 4:55:36 PM

Roles

Roles:	None

The figure shows the overall architecture for this application.

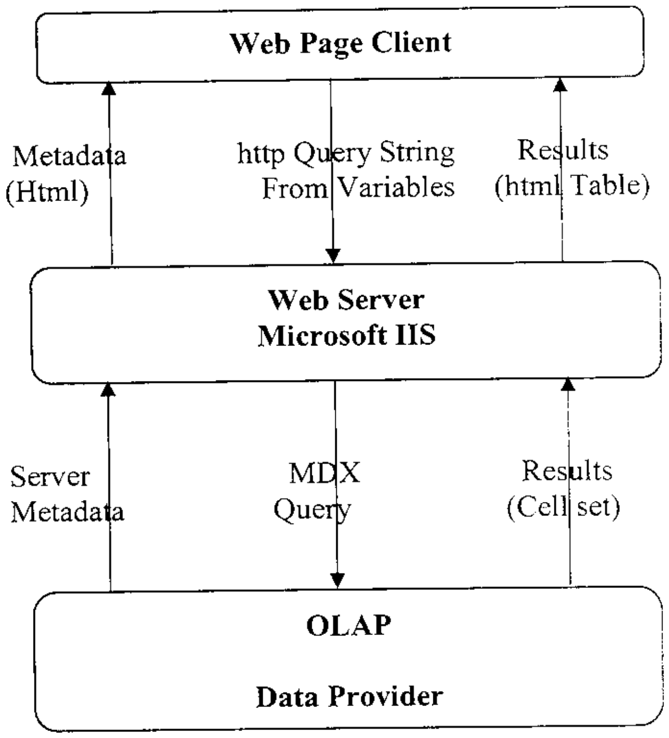


The Web page client, the Web Server, and the OLAP data provider, make up the three tiers of the OLAP Web Client

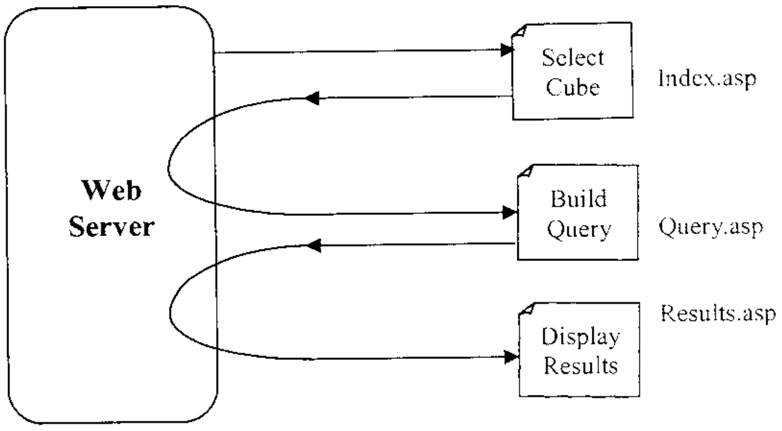
The client does not directly interface with the OLAP data provider. All interaction with the data server is conducted through the Web Server. By defining the architecture with this restriction, there is no need to establish an ADO connection between the client Web page and the data provider.

This approach has some advantages. In a security perspective, all users assume the Windows NT identity of the Internet Guest Account. This simplifies the need to manage access to the data provider.

The primary data interfaces between the tiers of the Web application is shown below. The complete OLAP Web application requires a series of sequential transactions between the three tiers.



To provide a more structured presentation of the Web client, the user interaction is broken down into three sequential Web pages that collect and present the information necessary to execute the query. The structured figure shown below provides a simplified function representation of the relationship between the Web pages, the Web server, and the user.



Developing the Web Application:

As shown in the structured figure the user interacts with three Web pages to retrieve data from the OLAP data provider. Each of these pages is discussed in subsequent subsections.

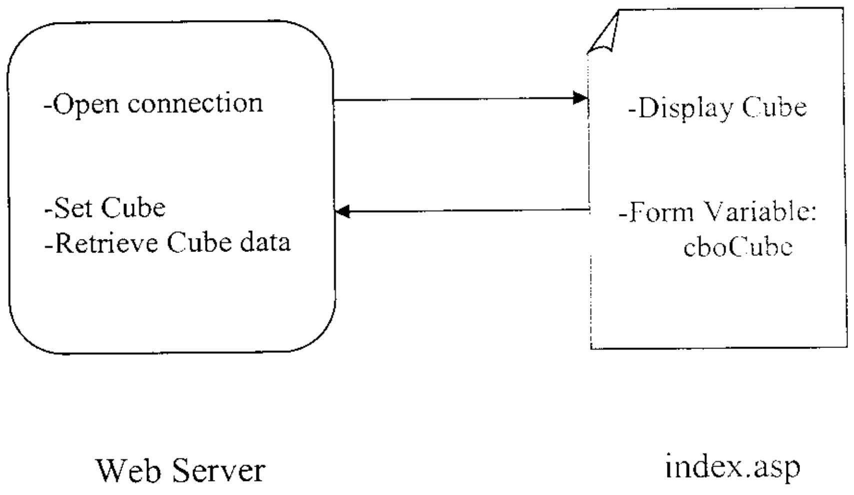
Connecting to the Server and Selecting the Cube

The first overall step in building OLAP Web client is

- Establish a connection with the OLAP data provider.
- Setting the database.
- Providing a list of cubes from which the user can select.

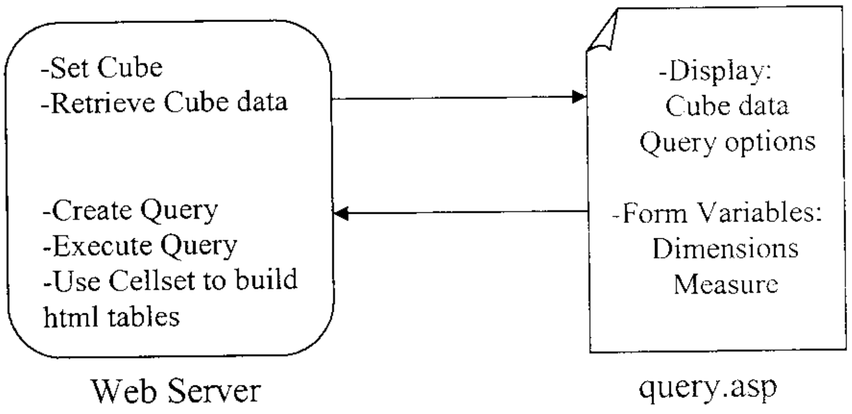
The specific approaches to each of these functions are as follows:

- The Web server establishes the connection to the OLAP data source using the Internet Guest Account. No user interaction is required.
- The Web server determines the database in the data source that will be exposed. No user interaction is required.
- The Web server creates a Web page that contains an html form prompting the user to select a cube.
- The user submits the form with the name of the cube to the server.



Displaying the Metadata to Select the Query Parameters

As specified in the form tag, the variables submitted by the index.asp page are processed by query.asp, as shown below.



The query.asp has three sections.

1st Segment

- Contains the standard html header information and the code that creates the cube definition object, based on the user cube selection on the index.asp page.
- This object is used throughout query.asp to retrieve dimension, level, members and children.

2nd Segment

- Builds the html interface allowing the user to view cube dimensions and levels and to select the axes for each of the dimensions.
- Since measures are a must, the column axis is avoided for the selection.

3rd Segment

- When the user completes his selection in query.asp and submits the form, one form variable for each dimension is submitted to the Web Server.

- Parsing this form and generating the appropriate MDX query is accomplished in the results.asp page.

Building the Query and Displaying the Results

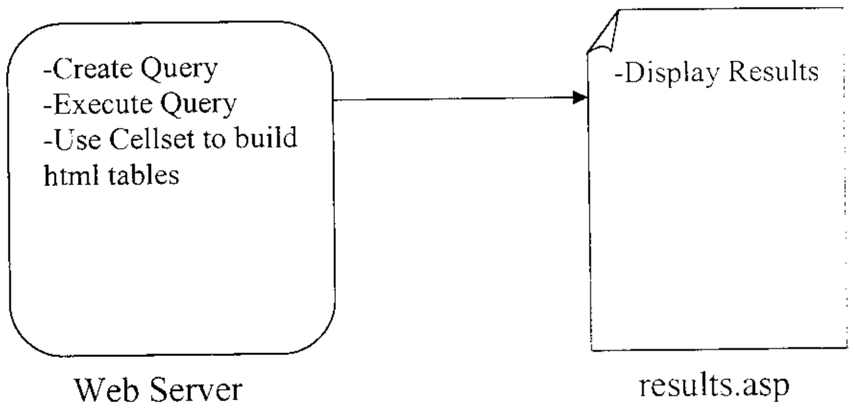
The entire cube's dimensions, all of the levels, all of the members and all of the children are exposed to the user. After the user has selected query parameters from this information, the parameters must be combined to produce an MDX query.

The MDX query is generated on the Web Server. This decision was made with the goals of a thin, simple client in mind and to take advantage of the power and stability of the server platform.

The query.asp page generates a number of form variables that are passed to the server with a standard submit action. It is the role of the Web Server, through the next ASP page (result.asp), to transform these variables into a query.

To complete the development of this application, it is necessary to

- Accept the form variables submitted in the query.asp
- Convert this information into an MDX query.
- Execute the query.
- Send the results back to the user.



The result.asp is divided into two sections.

1. Constructing the query.
2. Creating the result table.

The 1st section's code is to parse the form variables and construct an MDX query string. The iteration loop is made to retrieve all of the dimensions with the exception of the Measure dimension, which is passed in a second loop.

As each dimension is read it is placed in the row clause or the slicer clause. Since column clause is allotted for the measures. It should not be disturbed further.

When multiple joins are placed in the rows axes, they must be combined with a CrossJoin( ) function. Because a CrossJoin( ) function must be used for each different dimension, nested CrossJoins are likely in complex queries.

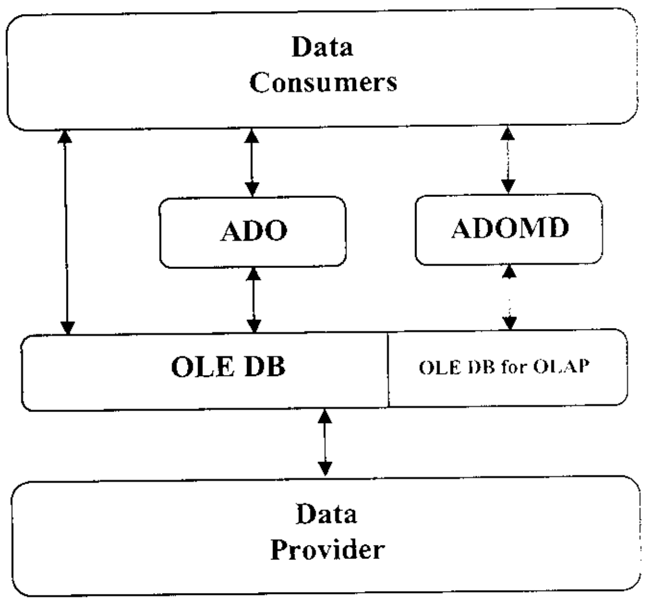
Combining dimensions in the slicers, or WHERE, clause is less complex because a comma separated list with the members or children level is the only requirement. Depending on the level count the .DefaultMember or children is used in the slicer clause.

When the column, row and slicer clauses are finished, they are combined into a complete MDX statement.

The 2nd code segment of result.asp executes the MDX query and translates the information from the resulting cellset object into an html table.

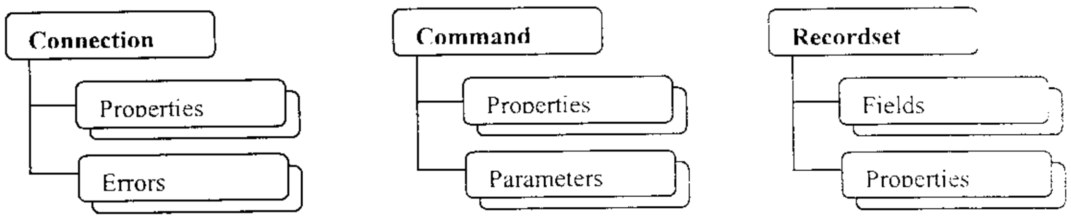
Before completing the explanation of the OLAP Web Client, it is better to know how the data is retrieved from the cellset.

There are two fundamental data connectivity layers that client applications can use to access data within Microsoft OLAP Services. OLE DB for OLAP extends OLE DB to include objects specific to multidimensional data, and ADOMD extends ADO in the same fashion.



The ActiveX Data objects Multidimensional Object Model

Concurrent with the development and release of OLAP Services, Microsoft updated its ADO object model. Specific extensions were added to the model to support the multidimensional data sources that were exposed through OLAP Services. The purpose and the structure of some of the most important objects – Connection, Command, and Recordset will be discussed with their collections.

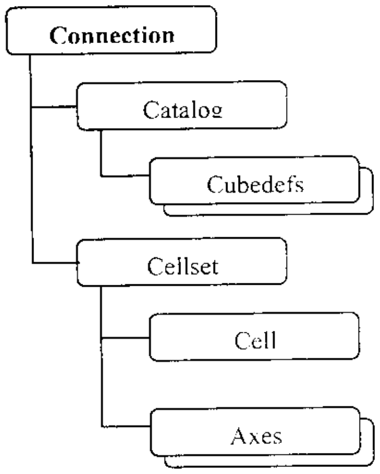


The Connection object consists of information about the data provider.

The Command object consists of information about command to be used on the Connection object.

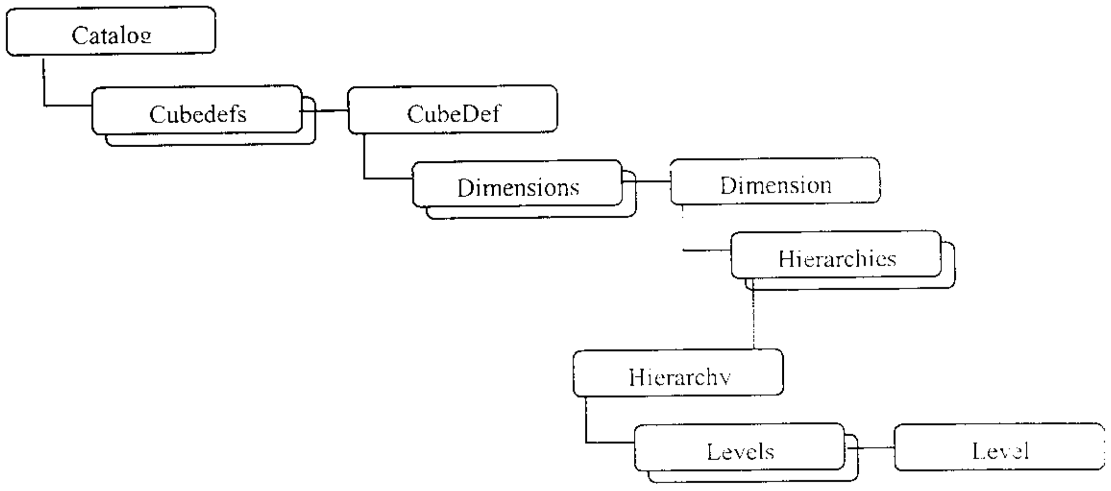
The Recordset object consists of the records returned from the execution of a query on the data provider.

The two objects within the Connection object critical, which are the multidimensional extensions to the basic ADO object model. The first object, the Catalog object, translates to a MD database hosted on the server. The second object, the Cellset object, contains the results of a query.



The Catalog Object

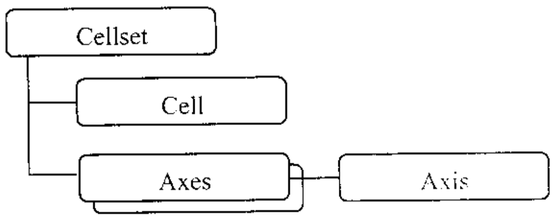
There are two practical uses for the Catalog object. First, in order to execute a query using a connection to the server, it is necessary to set the default database of the connection to the appropriate catalog. Second, access to the catalog object is the primary method of retrieving information about the data structure of the server.

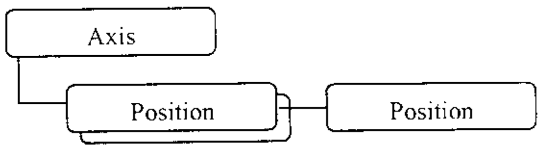


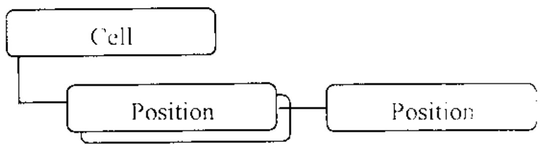
The Cellset Object

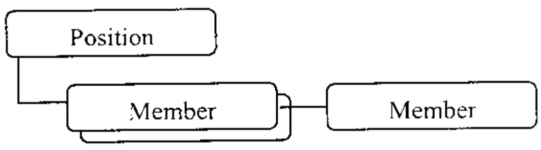
The Cellset object is the fundamental object for retrieving data itself. Like the recordset in ADO, the Cellset holds the result of an ADOMD query (MDX) after it has been executed.

The process populating the Cellset is simple. Specify the query and the connection, and execute the open method of the Cellset object. Because of the multidimensional nature of a cellset, the structure of a cellset is more complex than a recordset. The object hierarchy under the cellset is shown below.









Building an MDX query:

The most basic form, an MDX statement has a very simple composition-

SELECT *axis* [, *axis*]

FROM *cube*

WHERE *slicer* [, *slicer*]

The SELECT clause is used to define axis dimensions and the WHERE clause is used to supply slicer dimension.

The axis dimension consists of dimensions on the COLUMNS and dimensions on the ROWS. If more than one dimension has to be loaded in the COLUMNS or ROWS then CrossJoin( ) is used.

The FROM clause gets the name of the cube.

The WHERE clause consists of the dimensions separated by commas. The slicer dimensions contain the single members with which the cube is filtered or "sliced".

Some Examples:

SELECT {[Measures].[MeasuresLevel].Members} ON COLUMNS,
CrossJoin({[CM].Children},{[TIME].Children}) ON ROWS

FROM [RIMS MODEL]

WHERE ([PORTFOLIO].DefaultMember,[REGION].[All REGION].[North America])

The COLUMNS consists of all the members from the [Measures] as dimension with [MeasuresLevel] as hierarchy.

The ROWS consists of a combination of [CM] dimension and [TIME] dimension with all its immediate hierarchy.

The FROM specifies that the data has to be fetched from [RIMS MODEL] cube.

The WHERE contains the remaining dimensions in it. The [PORTFOLIO] dimension with all its default members is selected with [REGION] as dimension, [All REGION] as hierarchy, [North America] as level for the Region dimension and they are separated by comma.

```
SELECT {[Measures].[MeasuresLevel].Members} ON COLUMNS,
{[PORTFOLIO].Children} ON ROWS

 FROM [RIMS MODEL]

 WHERE ([CM].[All CM].[Technology].[Software].

            [REGION].[All REGION].[North America].[United States].

            [TIME].[All TIME].[2002].[2].[5])
```

The COLUMNS consists of all the members from the [Measures] as dimension with [MeasuresLevel] as hierarchy.

The ROWS consists of [PORTFOLIO] as dimension with all its immediate hierarchy known as Children.

The FROM specifies that the data has to be fetched from [RIMS MODEL] cube.

The WHERE contains the remaining dimensions in it. The [CM] as dimension, [All CM] as hierarchy, [Technology] as level, [Software] as member is selected with [REGION] as dimension, [All REGION] as hierarchy, [North America] as level, [United States] as member is selected with [TIME] as dimension, [All TIME] as hierarchy, [2002] as level, [2] as member, [5] as Child for the Time dimension and they are separated by comma.

Thus it is clear that the WHERE clause contains only specific dimensions like

[TIME].[All TIME].[2002].[2].[5]

[CM].[All CM].[Technology].[Software]

[PORTFOLIO].DefaultMember

It does not contain dimensions like

[PORTFOLIO].Children

[Measures].[MeasuresLevel].Members

4.5 OUTPUT DESIGN

OLAP RIMS Express index.asp page

Retrieves the cube from the OLAP data source.

SL. No.	Element Name	Description
1.	Select a cube	Caption
2.	List Box	Provides the list of cubes
3.	Button	Submit button used to send the information to the next page

query.asp page

Retrieves the dimension information according to the selected cube.

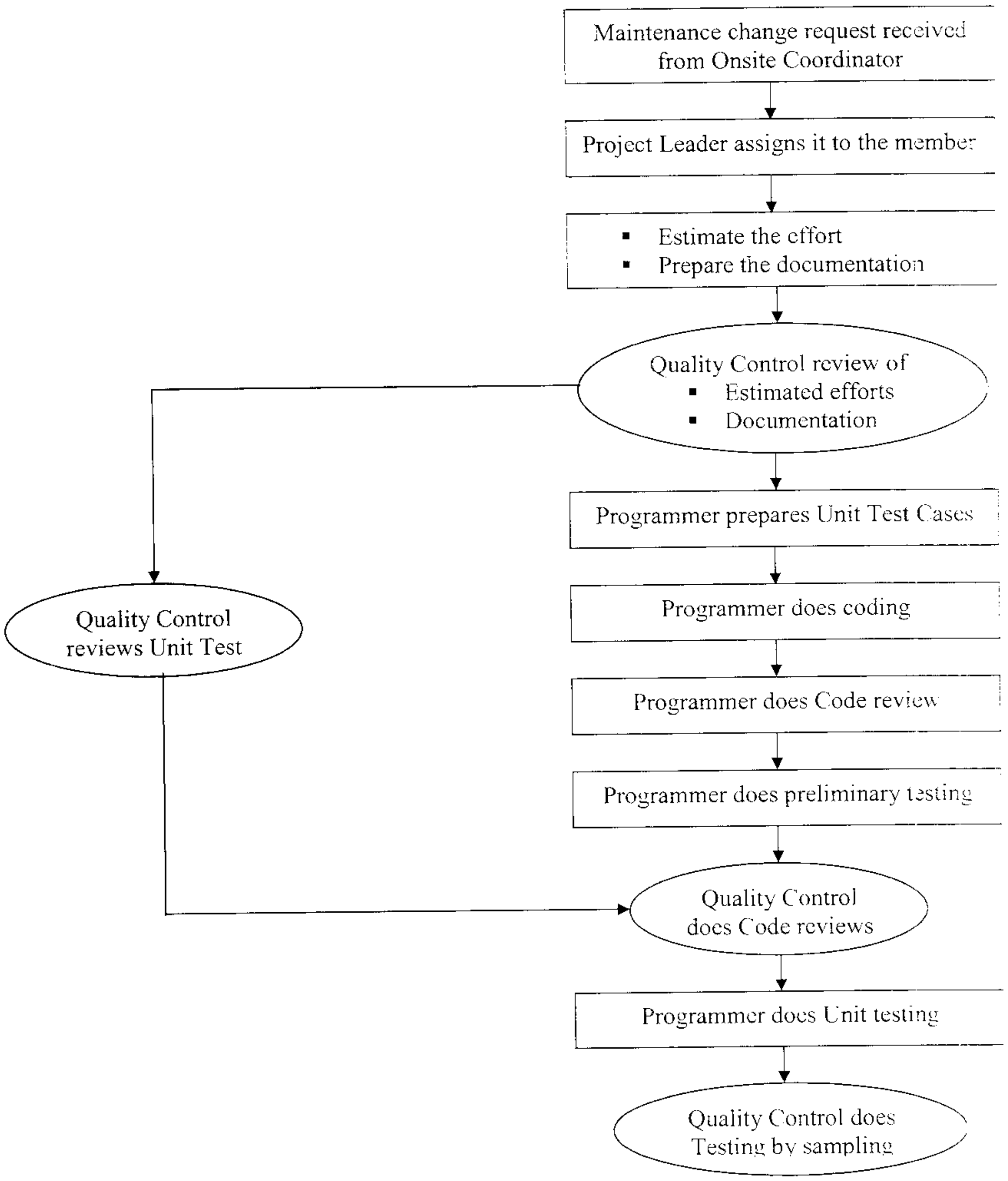
SL. No.	Element Name	Description
1.	Cube Name	Caption, specifies which cube is selected
2.	Dimension	Caption, categorizes the column as dimension
3.	Level	Caption, categorizes the column as level
4.	Axis	Caption, categorizes the column as axis
5.	CM	Caption, specifies the dimension
6.	PORTFOLIO	Caption, specifies the dimension
7.	REGION	Caption, specifies the dimension
8.	TIME	Caption, specifies the dimension
9.	MEASURES	Caption, specifies the dimension
10.	List Box	Provides the CM dimension in hierarchical structure
11.	List Box	Provides the PORTFOLIO dimension in hierarchical structure
12.	List Box	Provides the REGION dimension in hierarchical structure
13.	List Box	Provides the TIME dimension in hierarchical structure
14.	List Box	Provides the axis for the CM dimension
15.	List Box	Provides the axis for the PORTFOLIO dimension
16.	List Box	Provides the axis for the REGION dimension
17.	List Box	Provides the axis for the TIME dimension
18.	Button	Submit button used to send the information to the next page

result.asp page Displays the result

SL. No.	Element Name	Description
1.	Row Header Description	Caption, specifies the details about the row headers in the result
2.	CM DIMENISON	Caption, specifies the CM DIMENISON
3.	PORTFOLIO DIMENISON	Caption, specifies the PORTFOLIO DIMENISON
4.	REGION DIMENISON	Caption, specifies the REGION DIMENISON
5.	TIME DIMENISON	Caption, specifies the TIME DIMENISON
6.	Results	Caption, specifies the result
7.	Table	Provides the result of the selected query
8.	Query	Caption, specifies the result
9.	Text Box	Provides the query that is generated

SYSTEM IMPLEMENTATION & TESTING

5.1 SYSTEM IMPLEMENTATION AND TESTING


Maintenance change request received from Onsite Coordinator
Project Leader assigns it to the member
▪ Estimate the effort
▪ Prepare the documentation
Quality Control review of
▪ Estimated efforts
▪ Documentation
Programmer prepares Unit Test Cases
Programmer does coding
Programmer does Code review
Programmer does preliminary testing
Quality Control reviews Unit Test
Quality Control does Code reviews
Programmer does Unit testing
Quality Control does Testing by sampling

5.2 UNIT TEST PLAN

< name of the page>

#	Date	Version #	Section / Page # changed	Details of changes made
1				

Test Cycle Number:
Number of cases failed:
Number of cases passed:
Number of test cases added:
Number of test cases that cannot be tested at offshore:

Test Date:
Name of Tester:

Case #	Test Case	Procedure	Expected Results	Results	
				Pass/Fail	Comments
1.					

5.3 SYSTEM TESTING

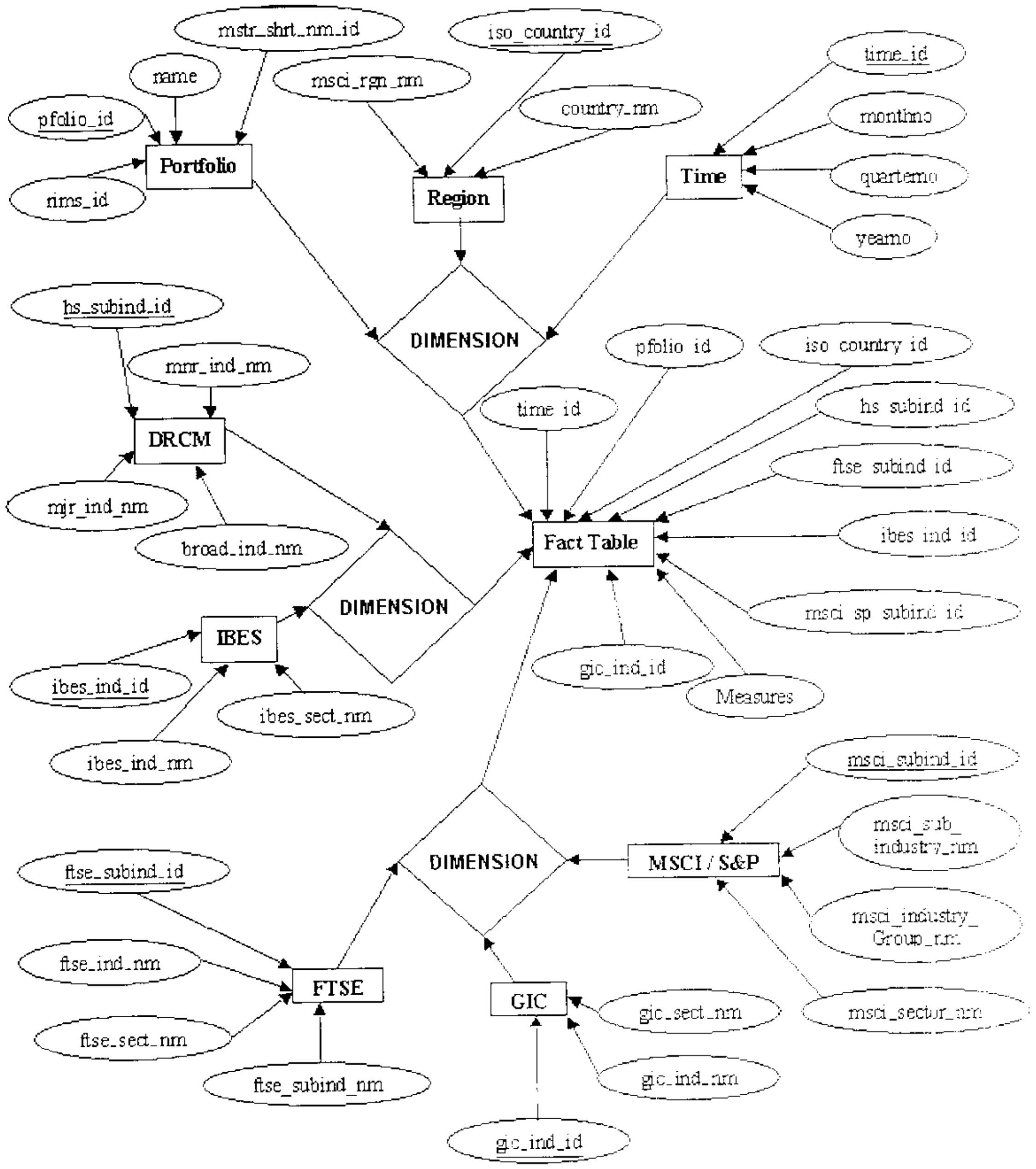
Test Cycle Number: 1
Number of cases failed: 0
Number of cases passed: 8
Number of test cases added:
Number of test cases that cannot be tested at offshore: 0

Test Date: 13 Mar 2002
Name of Tester: Raghunandan

Case #	Test Case	Procedure	Expected Results	Results	
				Pass/Fail	Comments
1.	The connection to the data source and the catalog.	Test for loading the Connection page	The connection should display an empty page.	Pass	
2.	To display all the available cubes in the list box.	Test for loading the index page	The path of connection.asp file has to be present in the index.asp page. If path is available it gets the cube from the catalog.	Pass	
3.	To display the query page.	Click the 'Submit' button	Opening page query.asp	Pass	
4.	To load all the elements and their respective values of query.asp page.	Test manually with the cube dimensions	All the values from the cube's dimensions should present in a hierarchical manner. The Axis column's List box should contain Slicer and Row.	Pass	
5.	To display the result page.	Click the 'Submit' button	Opening page result.asp	Pass	
6.	To load the result.asp page.	Test manually with the cube browsers result for the same query	The Row header description displays the selected dimension. The Result displays the values in a tabular format. The Query displays the MDX query for the selected dimensions. The time for loading the page is also displayed	Pass	

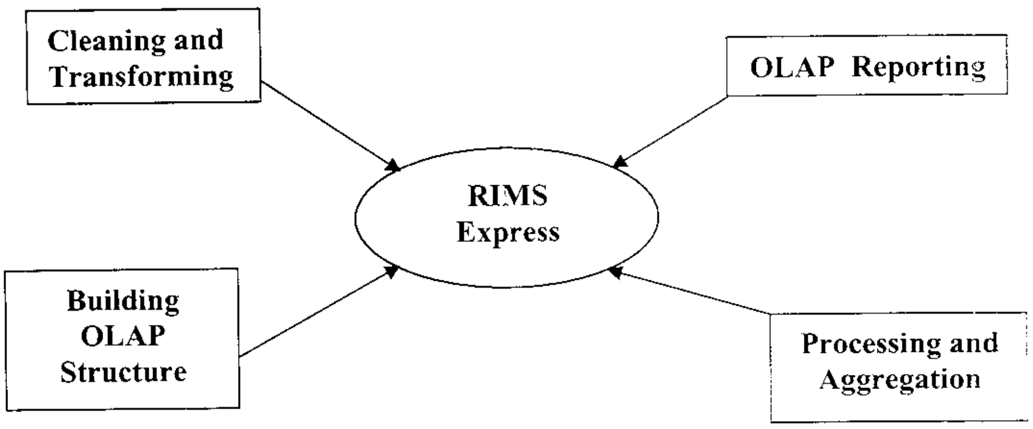
Case #	Test Case	Procedure	Expected Results	Results	
				Pass/Fail	Comments
7.	To check the performance **Configuration** **CPU: P4 1.2 GHz** **RAM: 256** **Configuration** **CPU: P3 1.6 GHz** **RAM: 128**	Selected dimensions CM: CM-All CM-Commercial Goods & Services-Capital Goods-Electrical Equipment Portfolio: All Region: All Time: Time-All Time-2002-2-5	OLAP Time Difference in Seconds is 41 RDBMS Time Difference in Seconds is 81 On an average there is 50% performance improvement for OLAP over RDBMS OLAP Time Difference in Seconds is 51 RDBMS Time Difference in Seconds is 94 On an average there is 50% performance improvement for OLAP over RDBMS	Pass	
8.	To check the performance **Configuration** **CPU: P4 1.2 GHz** **RAM: 256** **Configuration** **CPU: P3 1.6 GHz** **RAM: 128**	Selected dimensions CM: CM-All CM-Financials-Diversified Financials Portfolio: All Region: Region-All Region-Asia/Pacific Ex Japan-India Time: All	OLAP Time Difference in Seconds is 39 RDBMS Time Difference in Seconds is 81 On an average there is 50% performance improvement for OLAP over RDBMS OLAP Time Difference in Seconds is 35 RDBMS Time Difference in Seconds is 74 On an average there is 50% performance improvement for OLAP over RDBMS	Pass	

6.1 ENTITY RELATIONSHIP DIAGRAM

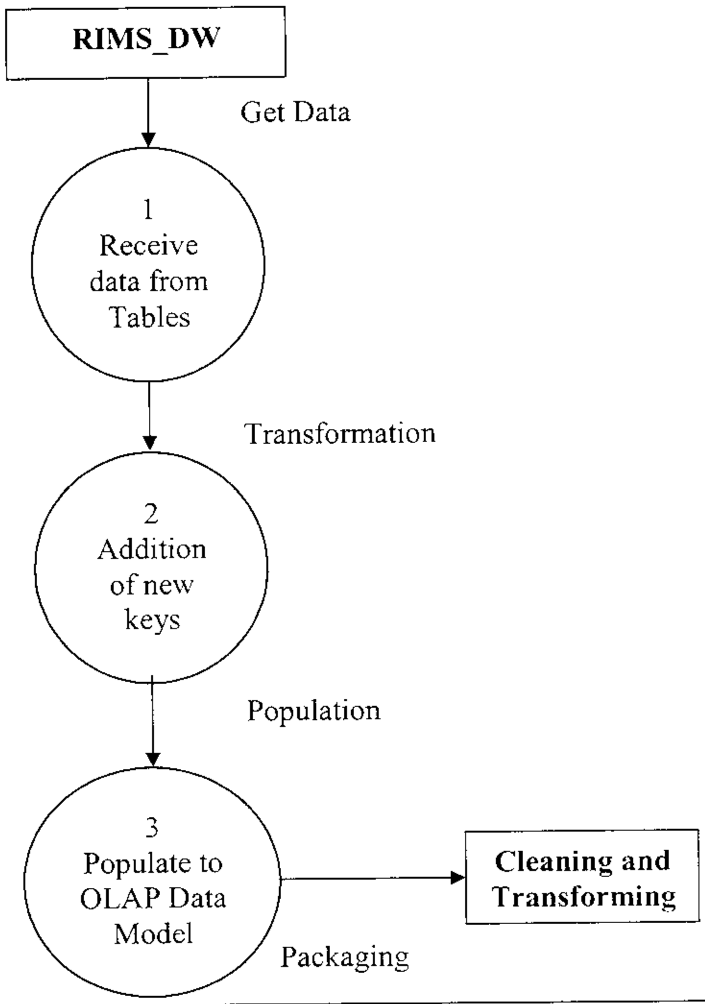


6.2 DATA FLOW DIAGRAM

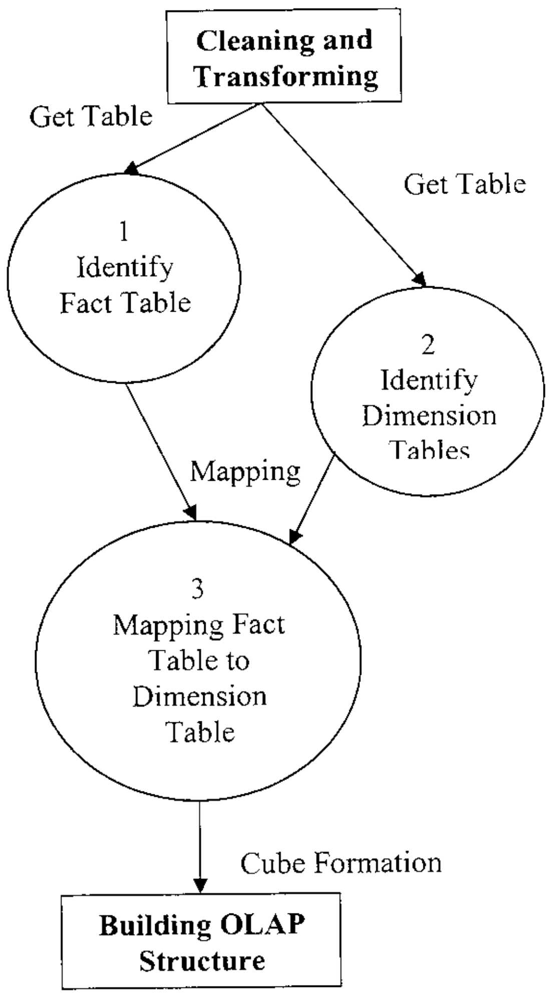
Level 0 DFD for RIMS Express



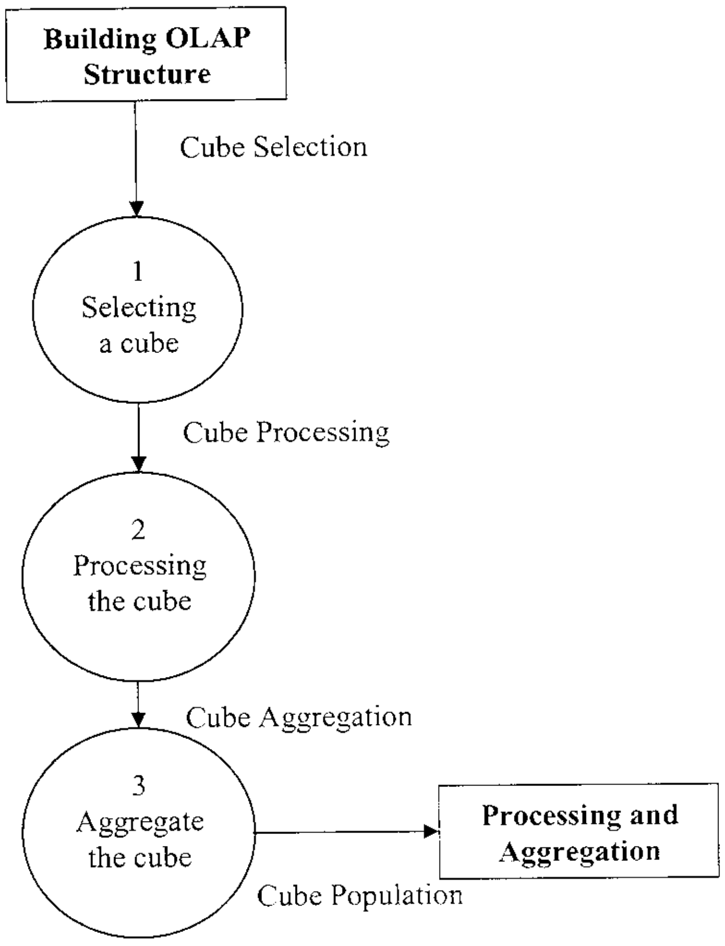
Level 1 DFD for Cleaning and Transforming


RIMS_DW
Get Data
1 Receive data from Tables
Transformation
2 Addition of new keys
Population
3 Populate to OLAP Data Model
Cleaning and Transforming
Packaging

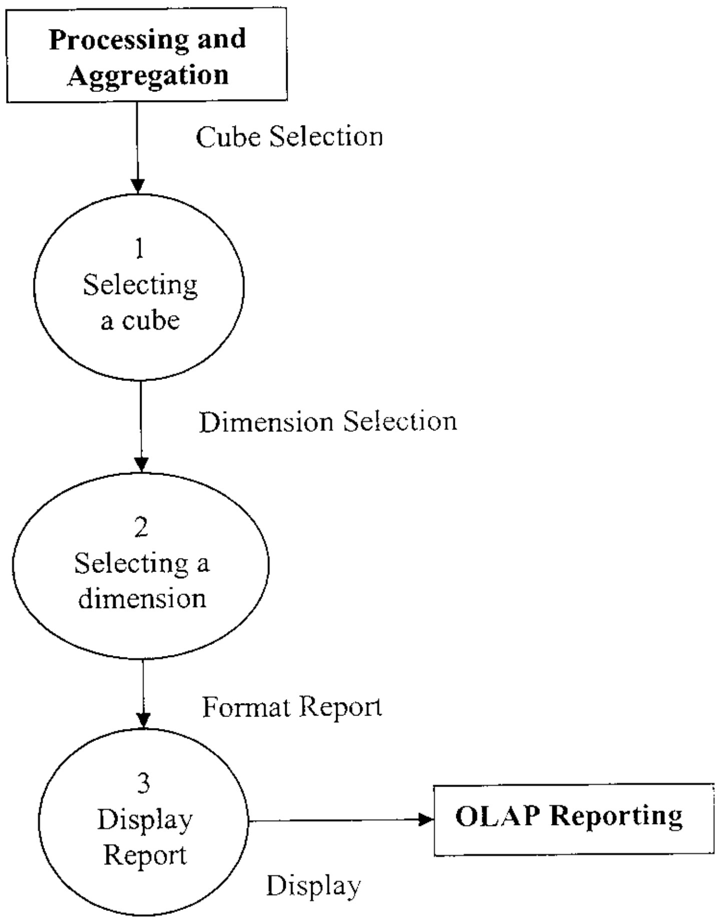
Level 1 DFD for Building OLAP Structure


Cleaning and Transforming
Get Table
Get Table
1 Identify Fact Table
2 Identify Dimension Tables
Mapping
3 Mapping Fact Table to Dimension Table
Cube Formation
Building OLAP Structure

Level 1 DFD for Processing and Aggregating

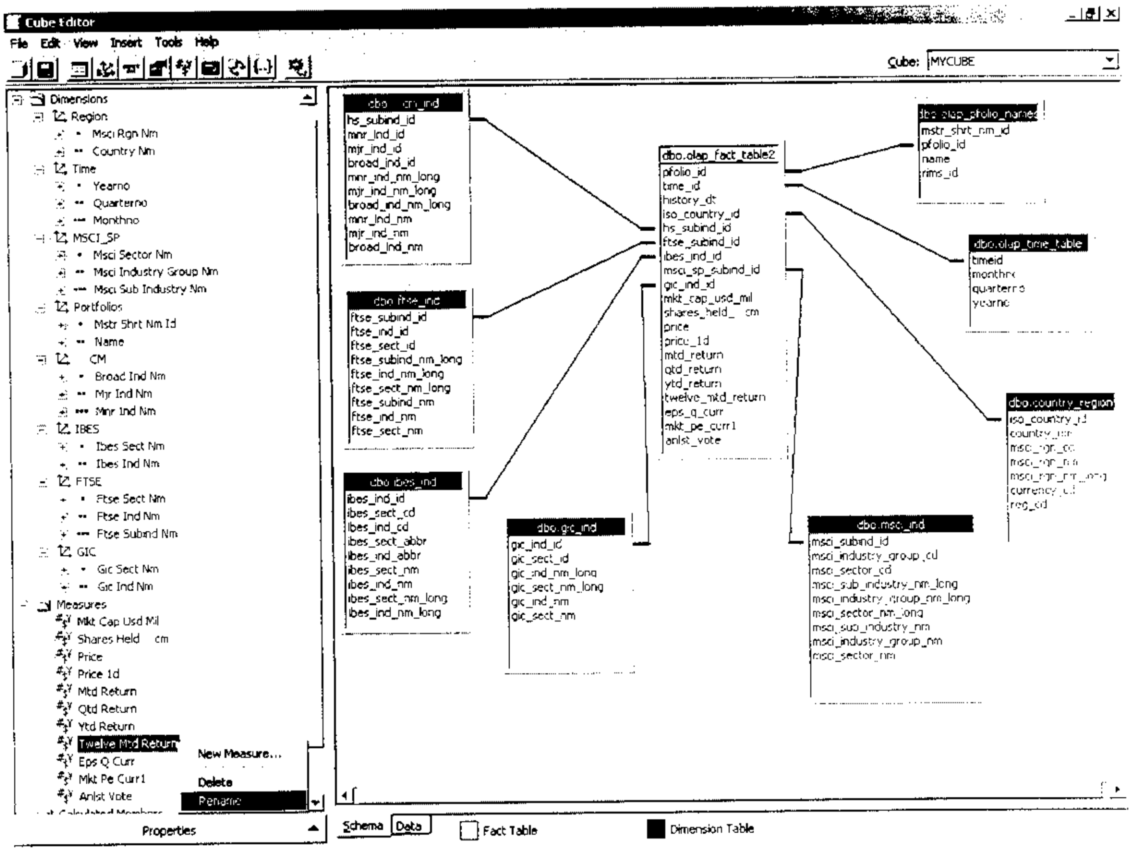


Level 1 DFD for OLAP Reporting

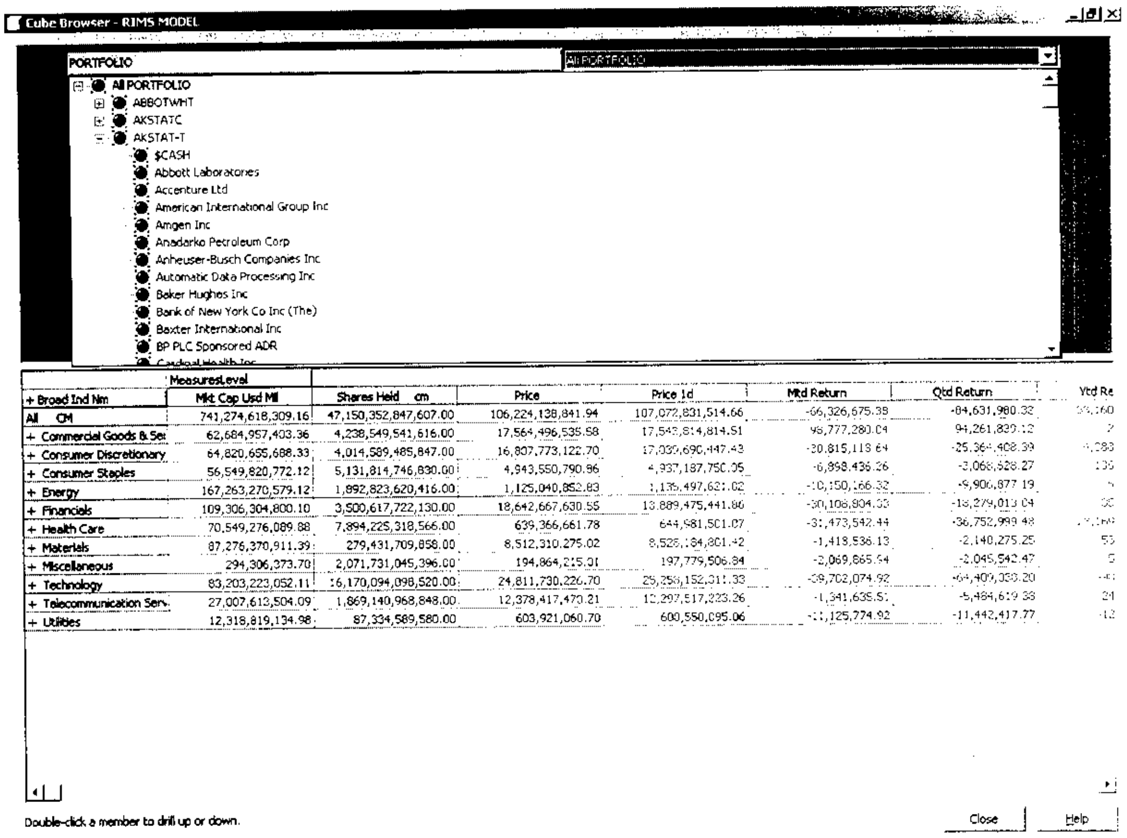


7. SCREEN LAYOUTS

The star schema representation of the cube is shown below. The left side pane contains the Dimensions and Measures. The Right side pane shows the Star schema with Fact Table surrounded by the dimension table. There is a Popup menu that shows that measure names can be easily changed.



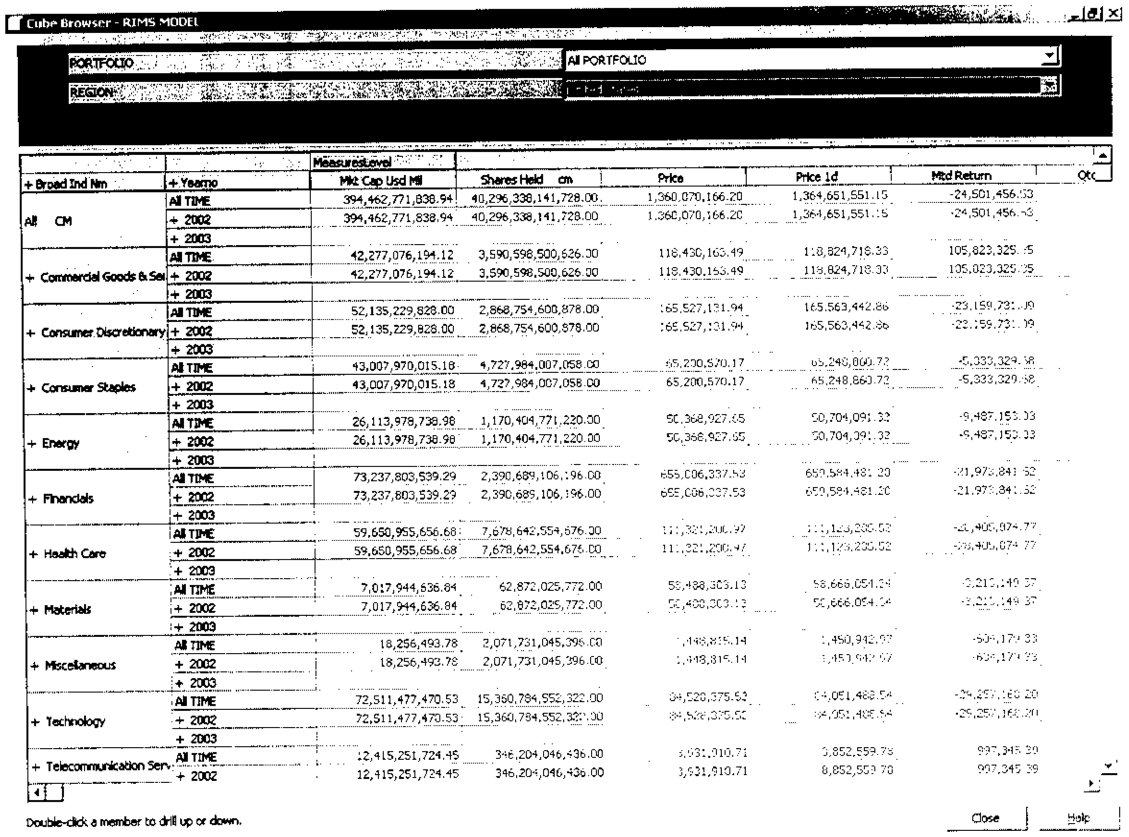
The Cube Browser gets the result from a specific cube. The browser shows all the dimensions that are present for the cube. The Dimension named PORTFOLIO with its hierarchy is shown below. The result pane shows the data for the selected dimension. Here CM Dimension and the Measures are selected.


Cube Browser - RIMS MODEL
PORTFOLIO
All PORTFOLIO
Double-click a member to drill up or down.
Close
Help

- All PORTFOLIO
  - ABBOTWHT
  - AKSTATC
  - AKSTAT-T
    - $CASH
    - Abbott Laboratories
    - Accenture Ltd
    - American International Group Inc
    - Amgen Inc
    - Anadarko Petroleum Corp
    - Anheuser-Busch Companies Inc
    - Automatic Data Processing Inc
    - Baker Hughes Inc
    - Bank of New York Co Inc (The)
    - Baxter International Inc
    - BP PLC Sponsored ADR
    - Cardinal Health Inc

	MeasuresLevel						
+ Broad Ind Nm	Mkt Cap Usd Mil	Shares Held cm	Price	Price 1d	Mtd Return	Qtd Return	Ytd Re
All CM	741,274,618,309.16	47,150,352,847,607.00	106,224,138,841.94	107,072,831,514.66	-66,326,675.38	-84,631,980.32	[illegible]
+ Commercial Goods & Ser	62,684,957,403.36	4,238,549,541,616.00	17,564,496,535.58	17,543,814,814.51	96,777,280.04	94,261,839.12	[illegible]
+ Consumer Discretionary	64,820,655,688.33	4,014,589,485,847.00	16,807,773,122.70	17,039,690,447.43	-30,815,113.64	-25,364,408.39	[illegible]
+ Consumer Staples	56,549,820,772.12	5,131,814,746,830.00	4,943,550,790.96	4,937,187,750.05	-6,898,436.26	-3,066,526.27	[illegible]
+ Energy	167,263,270,579.12	1,892,823,620,416.00	1,125,040,852.83	1,135,497,621.02	-10,150,166.32	-9,906,877.19	[illegible]
+ Financials	109,306,304,800.10	3,500,617,722,130.00	18,642,667,630.55	13,889,475,441.86	-30,106,804.33	-18,279,013.04	[illegible]
+ Health Care	70,549,276,089.88	7,894,225,318,566.00	639,366,661.78	644,981,501.07	-31,473,542.44	-36,752,999.48	[illegible]
+ Materials	87,276,370,911.39	279,431,709,858.00	8,512,310,275.02	8,526,184,801.42	-1,418,536.13	-2,140,275.25	55
+ Miscellaneous	294,306,373.70	2,071,731,045,396.00	194,864,215.01	197,779,506.84	-2,069,865.54	-2,045,542.47	[illegible]
+ Technology	83,203,223,052.11	16,170,094,098,520.00	24,811,730,226.70	25,256,152,311.33	-39,702,074.92	-64,409,038.20	[illegible]
+ Telecommunication Serv	27,007,613,504.09	1,869,140,968,848.00	12,378,417,470.21	12,297,517,223.26	-1,341,635.57	-5,464,619.38	21
+ Utilities	12,318,819,134.98	87,334,589,580.00	603,921,060.70	600,550,095.06	-11,125,774.92	-11,442,417.77	-12

This screen shows the CM dimension with the Time dimension for the measures. For every CM dimension member there will be a Time Dimension members. For Energy there is ALL TIME, 2002, 2003 values similarly we have for the other CM members.



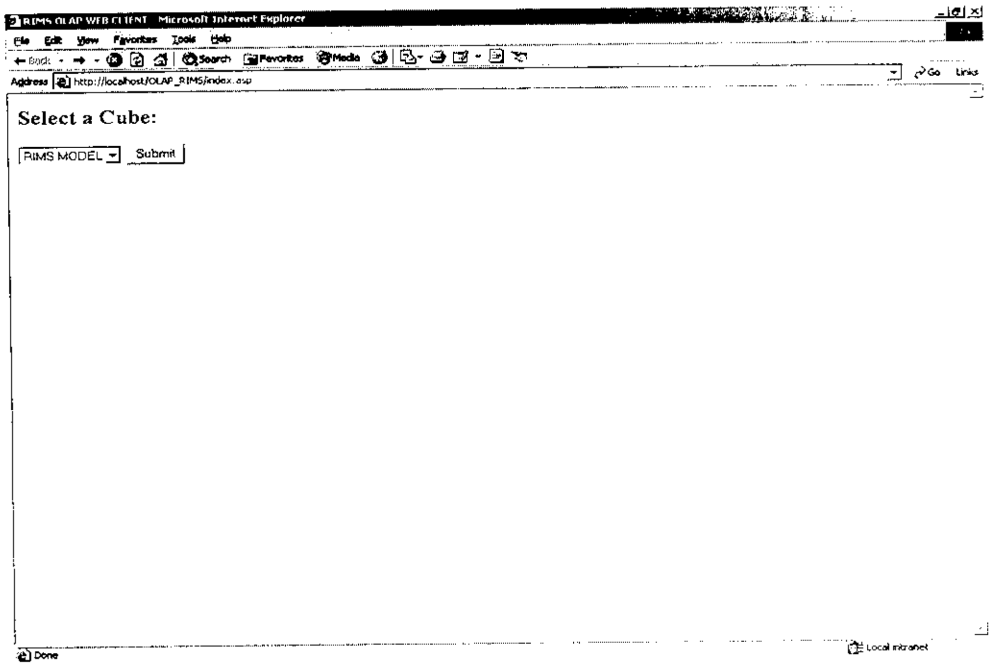
PORTFOLIO: All PORTFOLIO

REGION

+ Broad Ind Nm	+ Yearno	Mkt Cap Usd Mil	Shares Held cm	Price	Price 1d	Mtd Return
All CM	All TIME	394,462,771,838.94	40,296,338,141,728.00	1,360,070,166.20	1,364,651,551.15	-24,501,456.53
	+ 2002	394,462,771,838.94	40,296,338,141,728.00	1,360,070,166.20	1,364,651,551.15	-24,501,456.53
	+ 2003					
+ Commercial Goods & Ser	All TIME	42,277,076,194.12	3,590,598,500,626.00	118,430,163.49	118,824,718.33	105,823,325.45
	+ 2002	42,277,076,194.12	3,590,598,500,626.00	118,430,163.49	118,824,718.33	105,823,325.25
	+ 2003					
+ Consumer Discretionary	All TIME	52,135,229,828.00	2,868,754,600,878.00	165,527,131.94	165,563,442.86	-23,159,731.09
	+ 2002	52,135,229,828.00	2,868,754,600,878.00	165,527,131.94	165,563,442.86	-23,159,731.09
	+ 2003					
+ Consumer Staples	All TIME	43,007,970,015.18	4,727,984,007,058.00	65,200,570.17	65,248,860.72	-5,333,329.58
	+ 2002	43,007,970,015.18	4,727,984,007,058.00	65,200,570.17	65,248,860.72	-5,333,329.58
	+ 2003					
+ Energy	All TIME	26,113,978,738.98	1,170,404,771,220.00	50,368,927.65	50,704,091.32	-9,487,153.03
	+ 2002	26,113,978,738.98	1,170,404,771,220.00	50,368,927.65	50,704,091.32	-9,487,153.03
	+ 2003					
+ Financials	All TIME	73,237,803,539.29	2,390,689,106,196.00	655,006,337.53	650,584,481.20	-21,973,841.52
	+ 2002	73,237,803,539.29	2,390,689,106,196.00	655,006,337.53	650,584,481.20	-21,973,841.52
	+ 2003					
+ Health Care	All TIME	59,650,955,656.68	7,678,642,554,676.00	111,321,200.97	111,123,200.52	-28,405,074.77
	+ 2002	59,650,955,656.68	7,678,642,554,676.00	111,321,200.97	111,123,200.52	-28,405,074.77
	+ 2003					
+ Materials	All TIME	7,017,944,636.84	62,872,025,772.00	58,488,303.13	58,666,054.24	-3,210,149.37
	+ 2002	7,017,944,636.84	62,872,025,772.00	58,488,303.13	58,666,054.24	-3,210,149.37
	+ 2003					
+ Miscellaneous	All TIME	18,256,493.78	2,071,731,045,396.00	1,448,815.14	1,450,942.97	-604,179.33
	+ 2002	18,256,493.78	2,071,731,045,396.00	1,448,815.14	1,450,942.97	-604,179.33
	+ 2003					
+ Technology	All TIME	72,511,477,470.53	15,360,784,552,322.00	84,528,375.53	84,051,488.54	-29,257,160.20
	+ 2002	72,511,477,470.53	15,360,784,552,322.00	84,528,375.53	84,051,488.54	-29,257,160.20
	+ 2003					
+ Telecommunication Serv	All TIME	12,415,251,724.45	346,204,046,436.00	8,931,910.71	8,852,559.78	997,345.39
	+ 2002	12,415,251,724.45	346,204,046,436.00	8,931,910.71	8,852,559.70	997,345.39

Double-click a member to drill up or down.

This is the index page though which a cube is selected.

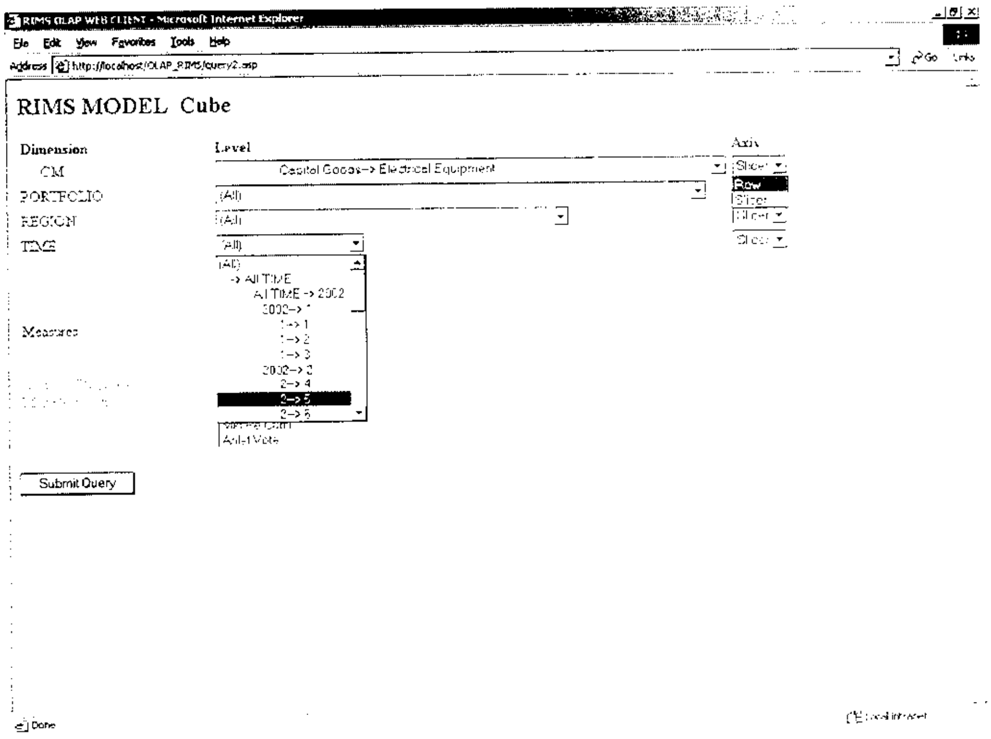

RIMS OLAP WEB CLIENT - Microsoft Internet Explorer

Address http://localhost/OLAP_RIMS/index.asp

Select a Cube:

RIMS MODEL | Submit

This is query page through which the Dimensions Levels and its Axis are chosen. The List Box shows the Hierarchy in the same format as the Cube Browser.


RIMS OLAP WEB CLIENT - Microsoft Internet Explorer

Address http://localhost/OLAP_RIMS/query2.asp

RIMS MODEL Cube

Dimension	Level	Axis
CM	Capital Goods--> Electrical Equipment	Slicer / Row / Slicer
PORTFOLIO	(All)	Slicer
REGION	(All)	Slicer
TIME	(All)	Slicer

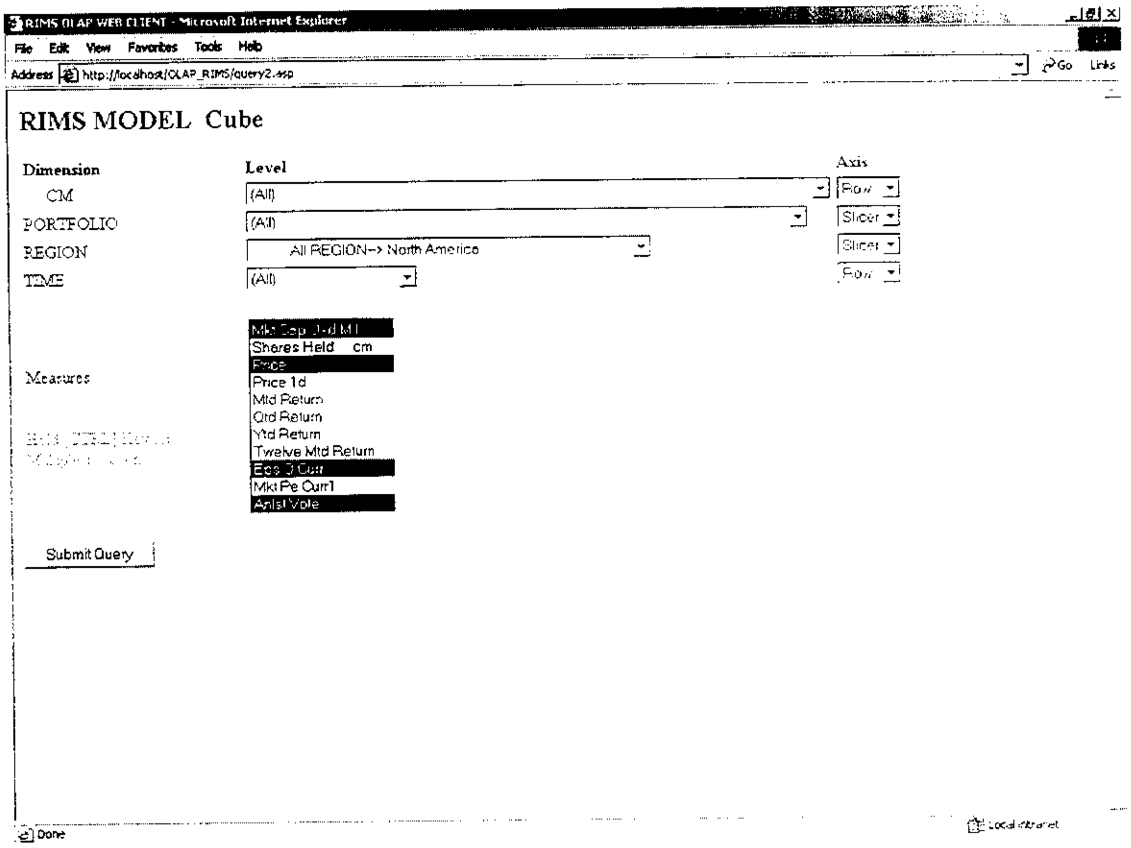
```
(All)
  -> All TIME
      All TIME -> 2002
       2002-> 1
         1-> 1
         1-> 2
         1-> 3
       2002-> 2
         2-> 4
         2-> 5
         2-> 6
```

Measures

Anti-1 Vote

Submit Query

This is the same query page, which shows how the Row and Slicer Axis are used. What ever level is selected in the Row Axis will be present in the result. Others that are in the Slicer Axis become data selection criteria. Here the Dimension REGION and PORTFOLIO are selected for the slicer Axis and the Dimension CM and TIME are selected for the row Axis.


RIMS OLAP WEB CLIENT - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address http://localhost/OLAP_RIMS/query2.asp

RIMS MODEL Cube

Dimension	Level	Axis
CM	(All)	Row
PORTFOLIO	(All)	Slicer
REGION	All REGION--> North America	Slicer
TIME	(All)	Row

Measures

Mkt Cap Usd Mil
Shares Held cm
Price
Price 1d
Mtd Return
Qtd Return
Ytd Return
Twelve Mtd Return
Eps 3 Curr
Mkt Pe Curr1
Anlst Vote

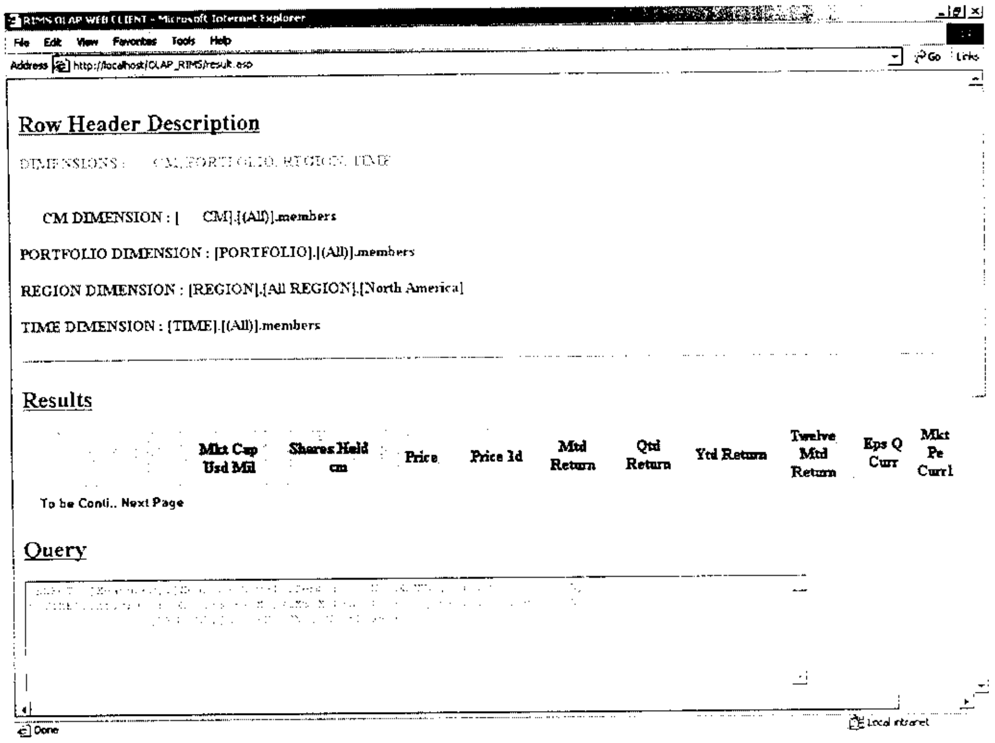
[illegible]

Submit Query

Done

Local intranet

This is the result page, which displays the selected dimension, the result and the MDX query for the result.


RIMS OLAP WEB CLIENT - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost/OLAP_RIMS/result.asp
Go
Links
Done
Local intranet

Row Header Description

DIMENSIONS : CM, PORTFOLIO, REGION, TIME

CM DIMENSION : [CM].[(All)].members

PORTFOLIO DIMENSION : [PORTFOLIO].[(All)].members

REGION DIMENSION : [REGION].[All REGION].[North America]

TIME DIMENSION : [TIME].[(All)].members

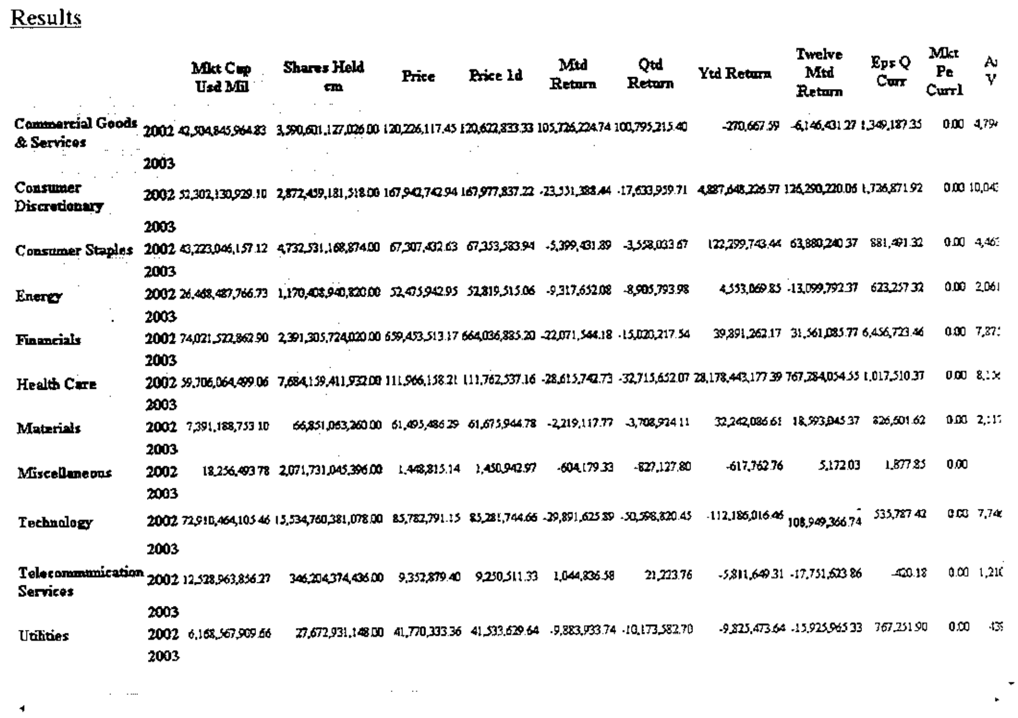
Results

Mkt Cap Usd Mil	Shares Held cm	Price	Price 1d	Mtd Return	Qtd Return	Ytd Return	Twelve Mtd Return	Eps Q Curr	Mkt Pe Curr1

To be Conti.. Next Page

Query

[illegible]



Results

		Mkt Cap Usd Mil	Shares Held cm	Price	Price 1d	Mtd Return	Qtd Return	Ytd Return	Twelve Mtd Return	Eps Q Curr	Mkt Pe Curr1	A V
Commercial Goods & Services	2002	42,504,845,964.83	3,590,601,127,026.00	120,226,117.45	120,622,833.33	105,726,224.74	100,795,215.40	-270,667.59	-6,146,431.27	1,349,187.35	0.00	4,79
	2003											
Consumer Discretionary	2002	52,302,130,929.10	2,872,459,181,518.00	167,942,742.94	167,977,837.22	-23,531,388.44	-17,633,959.71	4,887,648,226.97	126,290,220.06	1,726,871.92	0.00	10,04
	2003											
Consumer Staples	2002	43,223,046,157.12	4,732,531,168,874.00	67,307,432.63	67,353,583.94	-5,399,431.89	-3,558,033.67	122,299,743.44	63,880,240.37	881,491.32	0.00	4,46
	2003											
Energy	2002	26,468,487,766.73	1,170,408,940,820.00	52,475,942.95	52,819,515.06	-9,317,652.08	-8,905,793.98	4,553,069.85	-13,099,792.37	623,257.32	0.00	2,061
	2003											
Financials	2002	74,021,522,862.90	2,391,305,724,020.00	659,453,513.17	664,036,885.20	-22,071,544.18	-15,020,217.54	39,891,262.17	31,561,085.77	6,456,723.46	0.00	7,87
	2003											
Health Care	2002	59,706,064,499.06	7,684,159,411,932.00	111,966,158.21	111,762,537.16	-28,615,742.73	-32,715,652.07	28,178,443,177.39	767,284,054.55	1,017,510.37	0.00	8,1
	2003											
Materials	2002	7,391,188,753.10	66,851,063,260.00	61,495,486.29	61,675,944.78	-2,219,117.77	-3,708,924.11	32,242,086.61	18,593,845.37	826,501.62	0.00	2,1
	2003											
Miscellaneous	2002	18,256,493.78	2,071,731,045,396.00	1,448,815.14	1,450,942.97	-604,179.33	-827,127.80	-617,762.76	5,172.03	1,877.85	0.00	
	2003											
Technology	2002	72,910,464,105.46	15,534,760,381,078.00	85,782,791.15	85,281,744.66	-29,891,625.89	-50,598,820.45	-112,186,016.46	-108,949,366.74	535,787.42	0.00	7,74
	2003											
Telecommunication Services	2002	12,528,963,856.27	346,204,374,436.00	9,352,879.40	9,250,511.33	1,044,836.58	21,223.76	-5,811,649.31	-17,751,623.86	-420.18	0.00	1,21
	2003											
Utilities	2002	6,168,567,909.66	27,672,931,148.00	41,770,333.36	41,533,629.64	-9,883,933.74	-10,173,582.70	-9,825,473.64	-15,925,965.33	767,251.90	0.00	43
	2003											

8. CONCLUSION

OLAP RIMS EXPRESS the Prototype Model, which proves that OLAP reporting, is the best choice over RDBMS reporting for the existing system. There are around 500 Users for the existing RIMS EXPRESS system across the globe. This Model has the capability to attune with the existing architecture. The model will prove to yield high benefits when it is deployed on the web with more features.

9. SCOPE FOR FUTURE DEVELOPMENT

The OLAP RIMS EXPRESS is a three-tire architecture Prototype model. When this is deployed on the Web the application logic and processing will be focused on the middle tire of the architecture.

RDS (Remote Data Service) services can be used. Advanced Data Control, which is an invisible ActiveX component embedded within a web page will allow the page to communicate with an external datasource through any web server of choice.

The other service that can be used to leverage the environment is "disconnected recordsets", which facilitate the client-side data manipulation whereby the user can sort, filter and print data locally without ever having to re-request the data from the server. With this architecture, OLAP RIMS Express will be able to provide the flexibility to seamlessly integrate any type of content from almost any database, with complete location transparency.

By using specially written client-side binary components packaged as a dynamically downloadable COM Object, which can be used to enhance and optimize the data retrieval process. Sophisticated caching algorithms, security, and business logic residing in these components will provide enhanced capabilities such as pivoting, formatting, and aggregation.

10. BIBLIOGRAPHY

Reference Books

Fast Track to MDX	- Mark Whitehorn, Robert Zare and Mosha Pasumansky
Microsoft® SQL Server™ 2000 Resource Kit	- Microsoft Corporation
Microsoft OLAP Unleashed	- Peterson, et al
Programming Active Server Pages	- Scot Hillier, Daniel Mezick
The Comprehensive Guide to VBScript	- Richard Mansfield

Reference Web Sites

www.msdn.microsoft.com

www.asp.com

MDX

Calculated Members

Calculated members are members that are based not on data, but on evaluated expressions in MDX. They are returned in the same fashion as a normal member. MDX supplies a robust set of functions that can be used to create calculated members, giving extensive flexibility in the manipulation of multidimensional data.

User-Defined Functions

MDX provides extensibility in the form of user-defined functions using any programming language that can support Component Object Model (COM) interfaces. One creates and registers their own functions that operate on multidimensional data as well as accept arguments and return values in the MDX syntax. One can call user-defined functions from within Calculated Member Builder, data definition language (DDL) statements that support MDX, and MDX queries.

PivotTable Service

In Microsoft® SQL Server™ 2000 Analysis Services, MDX data definition and manipulation services are provided through PivotTable® Service. PivotTable Service also provides stand-alone OLE DB provider capabilities for multidimensional queries when not connected to an Analysis server. PivotTable Service is used for the definition and manipulation of local cubes, which can be used to locally store data in a multidimensional format.

Member Names and Member Keys

A member can be referenced by either its member name or by its member key. The previous example referenced the member by its member name, 4th quarter, in the Time dimension. However, the member name can be duplicated in the case of dimensions with nonunique member names, or it can be changed in the case of changing dimensions.

An alternate method to reference members is by referencing the member key. The member key is used by the dimension to specifically identify a given member. The ampersand (&) character is used in MDX to differentiate a member key from a member name, as shown in the following example:

```
[Time].[2nd half].&[Q4]
```

In this case, the member key of the 4th quarter member, Q4, is used. Referencing the member key ensures proper member identification in changing dimensions and in dimensions with nonunique member names.

The ampersand character can be used to indicate a member key reference in any MDX expression.

Calculated Members

Members can also be created, as part of an MDX query, to return data based on evaluated expressions instead of stored data in a cube to be queried. These members are called calculated members, and they provide a great deal of the power and flexibility of MDX. The WITH keyword is used in an MDX query to define a calculated member. For example, if one wants to provide a forecast estimate all of the packages by adding 10% of the existing value of the Packages measure, one can simply create a calculated member that provides the information and use it just like any other member in the cube, as demonstrated in the following example.

```
WITH MEMBER [Measures].[PackagesForecast] AS

'[Measures].[Packages] * 1.1'
```

Member Functions

MDX supplies a number of functions for retrieving members from other MDX entities, such as dimensions and levels, so that explicit references to a member are not always necessary. For example, the **FirstChild** function allows the retrieval of all the members from a given dimension or level; to get the first child member of the Time dimension, as demonstrated in the following example:

```
Time.[1st half]
```

One can also use the **FirstChild** function to return the same member, demonstrated in the next example.

```
Time.FirstChild
```

Set Functions

Explicitly typing tuples and enclosing them in braces is not the only way to retrieve a set. MDX supports a wide variety of functions that return sets.

The colon operator allows to use the natural order of members to create a set. For example, the following set:

```
{[1st quarter]:[4th quarter]}
```

retrieves the same set of members as the following set:

```
{[1st quarter], [2nd quarter], [3rd quarter], [4th quarter]}
```

The colon operator is an inclusive function; the members on both sides of the colon operator are included in the resulting set.

Other MDX functions that return sets can be used either by themselves or as part of a comma-delimited list of members. For example, all of the following MDX expressions are valid:

```
{Time.Children}
```

{Time.Children, Route.nonground.air}

{Time.Children, Route.nonground.air, Source.Children}

Sets and Dimensionality

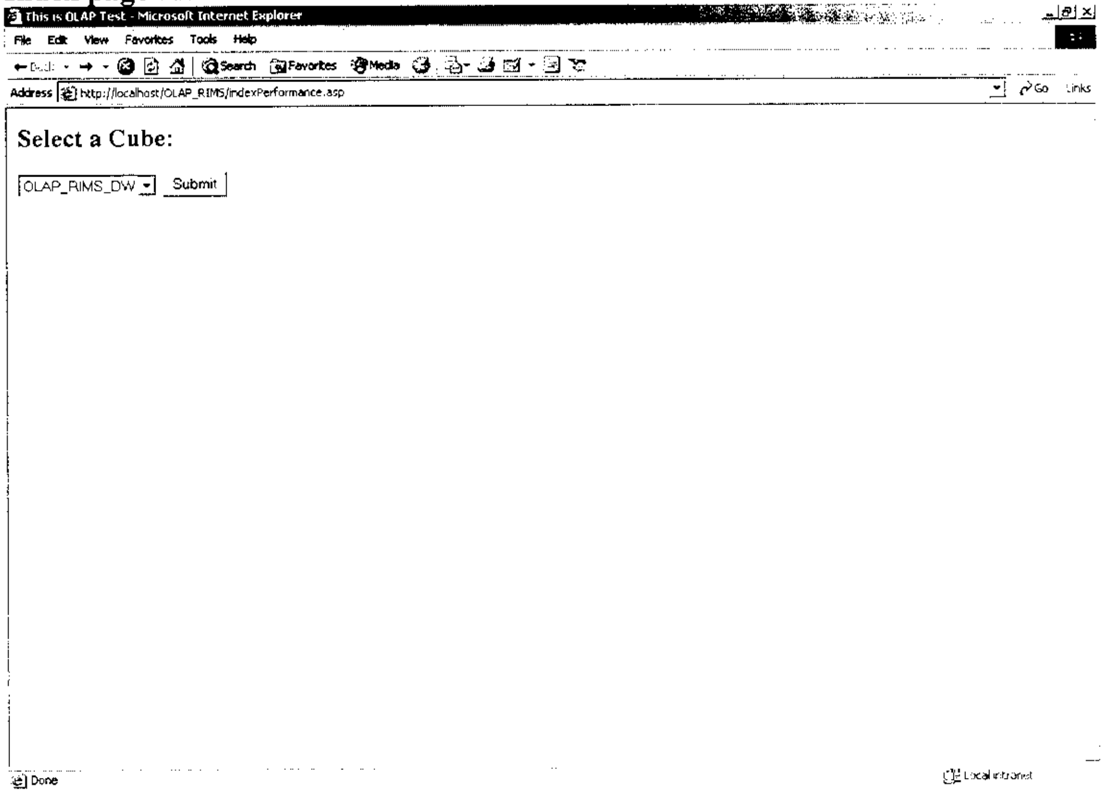
Like tuples, sets also have dimensionality. As a set is composed of tuples, so the dimensionality of a set is expressed by the dimensionality of each tuple within it. Because of this, tuples within a set must have the same dimensionality. In other words, this example would not work as a set:

{ (Time.[2nd half], Route.nonground.air), (Route.nonground.air, Time.[2nd half]) }

The order of tuples in a set is important; it affects, for example, the nesting order in an axis dimension. The first tuple represents the first, or outermost, dimension, the second tuple represents the next outermost dimension, and so on.

RDMBS Screen Layouts

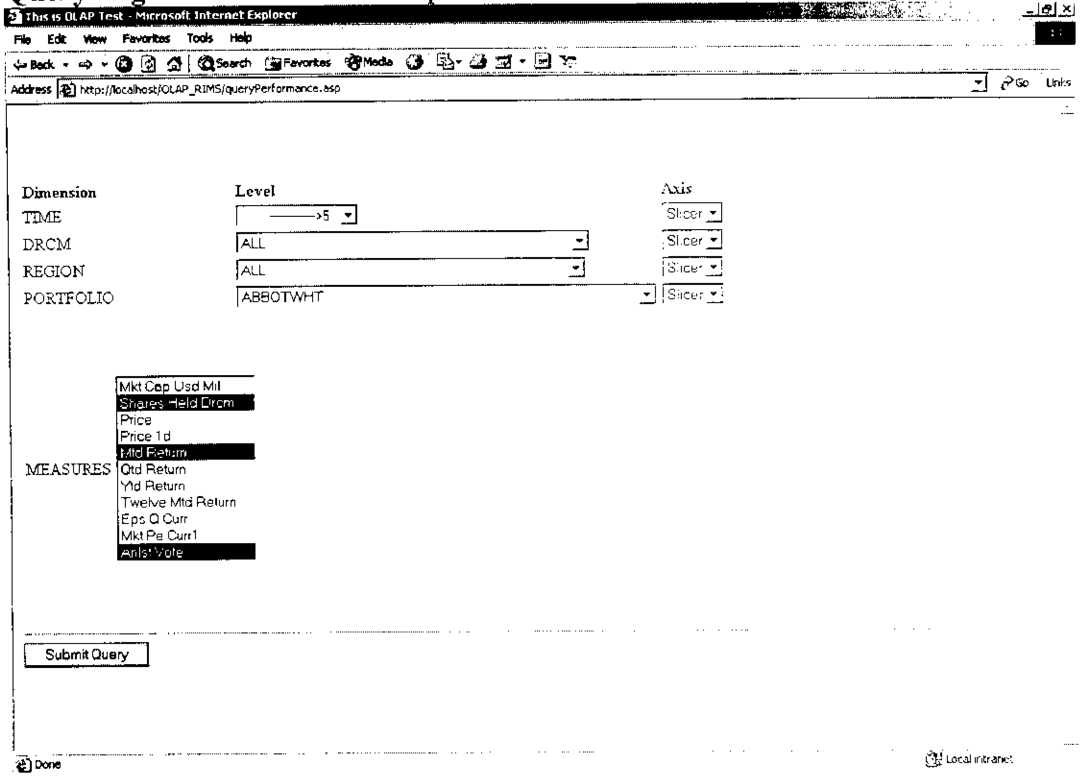
Index page : A database is selected as a cube


This is OLAP Test - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Search Favorites Media
Address http://localhost/OLAP_RIMS/indexPerformance.asp
Go Links
Done
Local intranet

Select a Cube:

OLAP_RIMS_DW Submit

Query Page : The Hierarchical representation of cube dimension is displayed.


This is OLAP Test - Microsoft Internet Explorer

Address http://localhost/OLAP_RIMS/queryPerformance.asp

Dimension	Level	Axis
TIME	———>5	Slicer
DRCM	ALL	Slicer
REGION	ALL	Slicer
PORTFOLIO	ABBOTWHT	Slicer

MEASURES

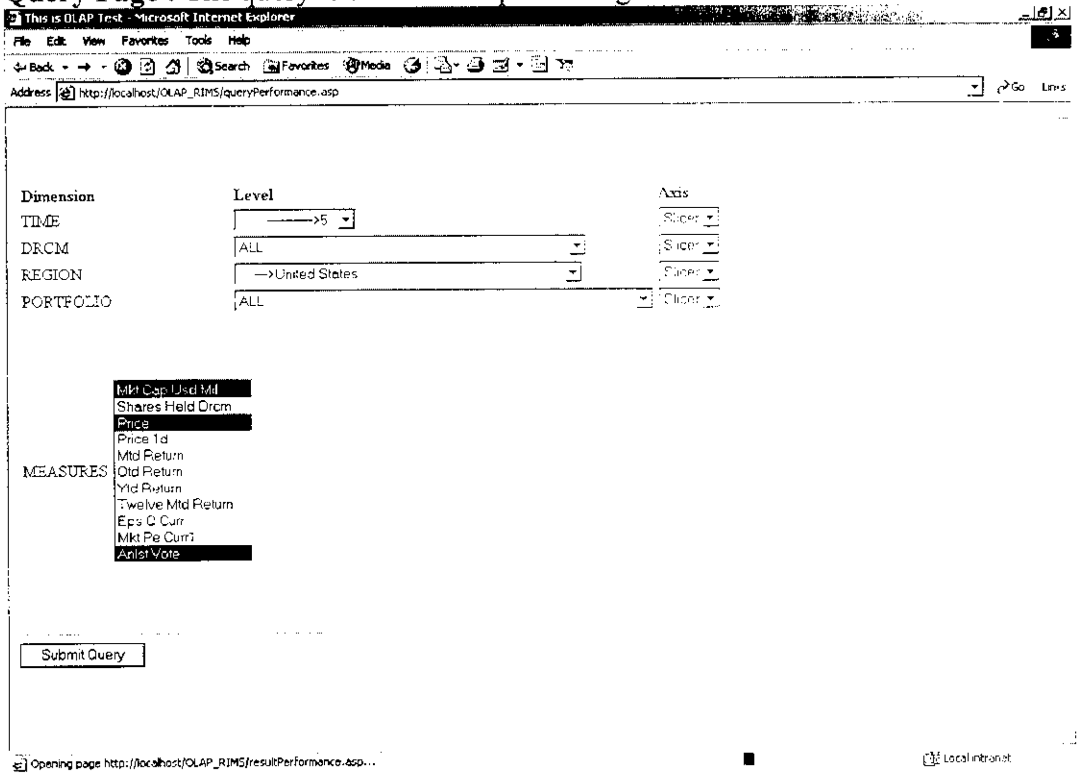
- Mkt Cap Usd Mil
- Shares Held Drcm
- Price
- Price 1d
- Mtd Return
- Qtd Return
- Ytd Return
- Twelve Mtd Return
- Eps Q Curr
- Mkt Pe Curr1
- Anlst Vote

Submit Query

Done

Local intranet

Query Page : The query is selected for processing.


This is OLAP Test - Microsoft Internet Explorer

Address http://localhost/OLAP_RIMS/queryPerformance.asp

Dimension	Level	Axis
TIME	———>5	Slicer
DRCM	ALL	Slicer
REGION	—>United States	Slicer
PORTFOLIO	ALL	Slicer

MEASURES

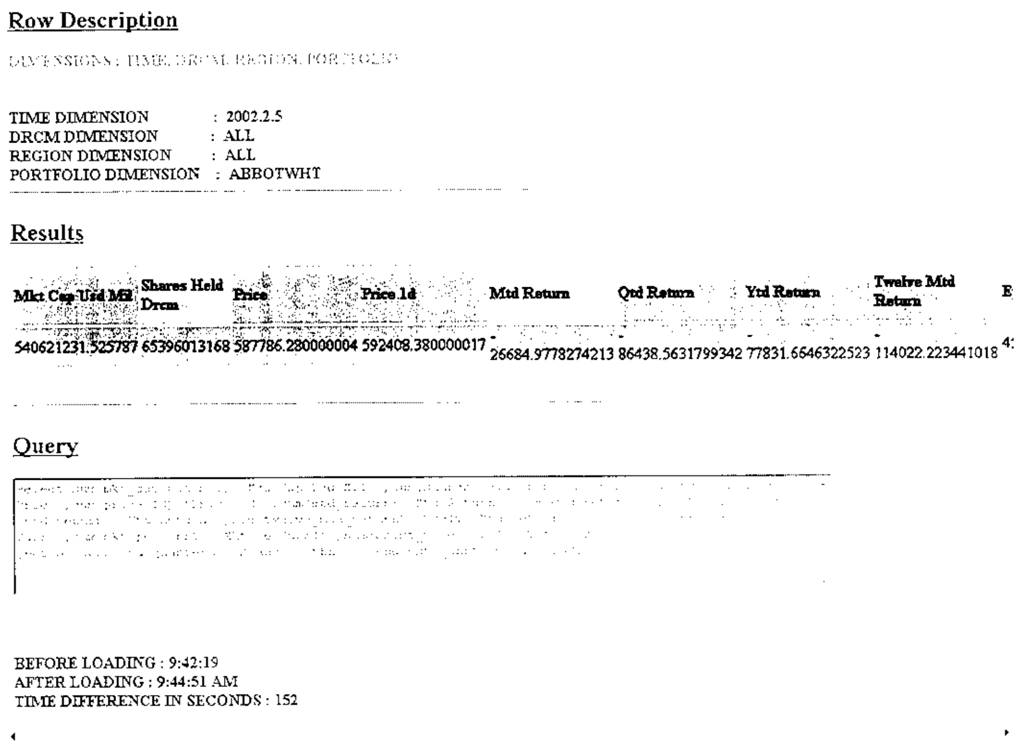
- Mkt Cap Usd Mil
- Shares Held Drcm
- Price
- Price 1d
- Mtd Return
- Qtd Return
- Ytd Return
- Twelve Mtd Return
- Eps Q Curr
- Mkt Pe Curr1
- Anlst Vote

Submit Query

Opening page http://localhost/OLAP_RIMS/resultPerformance.asp...

Local intranet

Result Page : The page displays the selected dimension, the sql query as well as the result.



Row Description

DIMENSIONS : TIME, DRCM, REGION, PORTFOLIO

TIME DIMENSION	:	2002.2.5
DRCM DIMENSION	:	ALL
REGION DIMENSION	:	ALL
PORTFOLIO DIMENSION	:	ABBOTWHT

Results

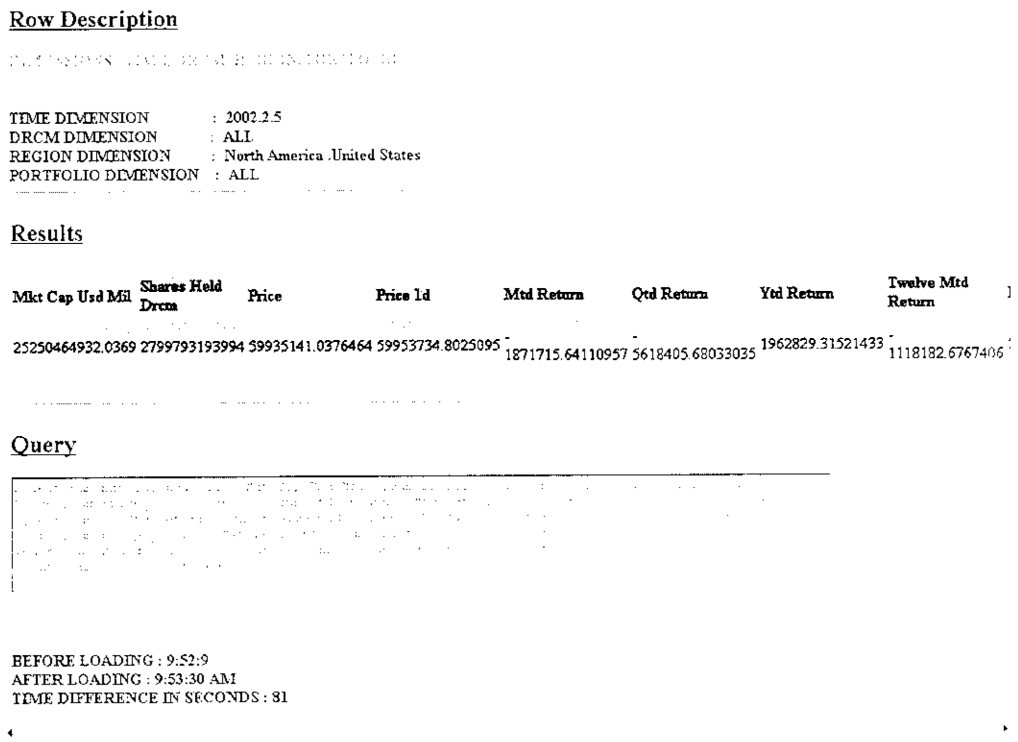
Mkt Cap Usd Mil	Shares Held Drcm	Price	Price 1d	Mtd Return	Qtd Return	Ytd Return	Twelve Mtd Return	E
540621231.525787	65396013168	587786.280000004	592408.380000017	-26684.9778274213	-86438.5631799342	-77831.6646322523	-114022.223441018	4:

Query

[illegible]

BEFORE LOADING : 9:42:19
AFTER LOADING : 9:44:51 AM
TIME DIFFERENCE IN SECONDS : 152

Result Page : The page is another proof of the result page.



Row Description

[illegible]

TIME DIMENSION	:	2002.2.5
DRCM DIMENSION	:	ALL
REGION DIMENSION	:	North America .United States
PORTFOLIO DIMENSION	:	ALL

Results

Mkt Cap Usd Mil	Shares Held Drcm	Price	Price 1d	Mtd Return	Qtd Return	Ytd Return	Twelve Mtd Return	]
25250464932.0369	2799793193994	59935141.0376464	59953734.8025095	-1871715.64110957	-5618405.68033035	1962829.31521433	-1118182.6767406	:

Query

[illegible]

BEFORE LOADING : 9:52:9
AFTER LOADING : 9:53:30 AM
TIME DIFFERENCE IN SECONDS : 81