

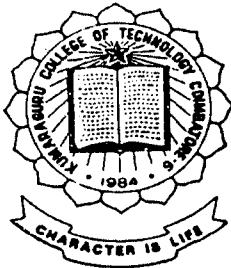
Resource Correlator

PROJECT REPORT

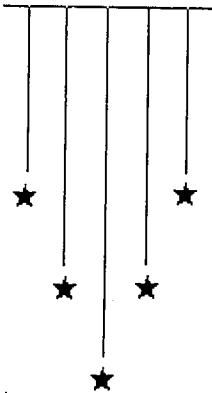
SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIRMENTS FOR THE
AWARD OF THE DEGREE OF

BACHELOR OF ENGINEERING

OF BHARATHIAR UNIVERSITY,
COIMBATORE.



1998 - 2002



P-662



Submitted By,

Arun R.,	9827K0162
Preethi S.,	9827K0198
Kavitha B.,	9827K0718
Manju Priyadarsini M.,	9827K0188

Guided By,

Ms.N.Rajathi,B.E.

DEPARTMENT OF COMPUTER SCINENCE & ENGINEERING
KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE – 641006.

MARCH 2002

Certificate



To whomsoever it may concern

Certified that this product "Resource Correlator 3" is the bonafide work of

Mr.R.Arun,

Ms.S.Preethi,

Ms.B.Kavitha,

Ms.M.ManjuPriyadarsini

who have carried out the project under our supervision. We certify further that to the best of our knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate. The documentation submitted by the group was well compiled and the project completely satisfied our expectations.

For Rushmore Consultancy Services (P) Ltd

V. Arul Kumar.
11/03/2002
Director

Moussali
Project Guid

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Kumaraguru College of Technology

(Affiliated to Bharathiar University, Coimbatore)

CERTIFICATE

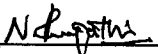
This is to Certify that project report entitled

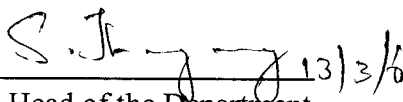
Resource Correlator

is a bonafide record of work done by

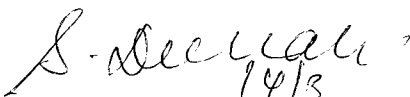
Arun R.,	9827K0162
Preethi S.,	9827K0198
Kavitha B.,	9827K0718
Manju Priyadarsini M.,	9827K0188

in partial fulfillment of the requirements for the award of the Degree of Bachelor of Engineering in Computer Science and Engineering branch of Bharathiar University, Coimbatore-641006 during the academic year 2001-2002.


Project Guide


Head of the Department

Submitted for the University Examination held on


Internal Examiner


External Examiner

*Dedicated to our
Beloved Parents*

Acknowledgement

Acknowledgement

We take this opportunity to express our gratitude to all, whose contribution in this project can never be forgotten.

We are extremely grateful to **Dr.K.K.Padmanabhan**, B.Sc. (Engg), M.Tech., Ph.D., Principal, Kumaraguru College of Technology for haven given us a golden opportunity to serve the purpose of our education.

We are deeply obliged to **Dr.S.Thangasamy**, Ph.D., Professor Head of Department of Computer Science and Engineering for his valuable guidance and useful suggestions during the course of this project.

We also extend our heartiest thanks to **Mrs.S.Devaki**, M.S., Assistant Professor, Department of Computer Science and Engineering for providing us her support which really helped us to come out with the project successfully.

Our heartfelt thanks to our project guide **Ms.N.Rajathi**, B.E., Lecturer, Department of Computer Science and Engineering for her constant guidance and unfailing interest in aiding us to consummate this project successfully.

We are indebted to our class advisor **Mrs.D.Chandrakala**, M.E., Lecturer, Department of Computer Science and Engineering for inspiring us and valuable support given to us throughout our project.

With immense pleasure, we express our heartfelt gratitude to **Mr.R.Rajkumar and Mr.V.Varatharaj Kumar**, the Managing Directors, Rushmore Consultancy Services (P) Ltd., Chennai for providing us the opportunity to do the project in their organization.

We also take this chance to equally express our gratitude to **Mr.V.Mouralidaran**, our project guide at the company for providing technical guidance and all the others of the organization for their constant motivation during the course of our project.

We extend our thanks and gratitude to all the non-teaching staffs of Computer Science and Engineering Department, Kumaraguru College of Technology for their best support and guidance provided to us.

Above all we owe our gratitude to our parents and friends for their support and God almighty for showering abundant blessings on us.

Synopsis

Synopsis

Resource Correlator 3 is a product used by MediaSeek Ltd. It is a system for creating, correlating and managing Educational Resource Data that is correlated to MediaSeek's Knowledge Base. It provides users with the ability to produce Correlation Matching Reports that show the matches between correlated educational resources and Guiding Documents.

Resource Correlator is a 32-bit Windows application, which is built on an *n-tier methodology* in which the different software chores are separated into software components. It consists of a middle tier, which is responsible for interaction between the data source and the user interface (UI) that the end-user sees when they are interacting with Resource Correlator.

The middle tier insulates the user interface from changes to the database-specific information it must process. With this middle tier in place, there is no requirement for the client side software to deal with ODBC particulars or any SQL code. This middle data access tier is provided in an ATL-based DLL called *Magnolia*.

Resource Correlator is intended to correlate resources in the K-12 education market. The target users of this product include Publishers, Independent educational consultants, Educators and Classroom teachers. The important feature, is the ability of other software projects within MediaSeek to utilize the Magnolia functionality for maintenance of Resource Correlator client-side data.

Contents

Contents

1. Introduction	
1.1 Company Profile	1
1.2 Existing System and its limitations	2
1.3 Proposed system and its advantages	3
2. System Requirement	
2.1 Product Definition	4
2.2 Project Plan	7
3. Software Requirement Specification	
3.1 Hardware Specifications	11
3.2 Software Specifications	11
3.3 Functional Specifications	20
4. Design Document	
4.1 Architectural Design Specifications	23
4.2 Database Structure	27
4.3 Table Design	28
5. Product Testing	32
6. Future Enhancement	34
7. Conclusion	35
8. Bibliography	36
Annexure	
1. Source Code	
2. User Interface	

Introduction

1. Introduction

This document outlines the functionality of Resource Correlator 3, a system for creating, correlating and managing Educational Resource Data that is correlated to Mediaseek's Knowledge Base. This system will consist of a Client and Internet browser-based functionality. Once the data is created it is submitted to Mediaseek for review and posting on Mediaseek's production database. The Resource Correlator system is one part of Mediaseek's Suite of Correlation tools, which also includes Guiding Document Correlator.

1.1 Company's Profile:

Rushmore Consultancy services (P) Ltd was conceived by a team of seasoned and dedicated professionals on June 03, 1999. It provides businesses with turnkey solutions that include software and services. Since inception, the company has been specializing in electronic commerce research & development, Internet Site Development and Custom Designed Business Software on n-tired business models.

Rushmore Global Services is composed of:

Consulting Services:

Applying proven project methodologies to successfully implement Rushmore solutions.

Education Services:

Education programs designed to ensure fully enabled and self-sufficient client sites.

Support Services

A range of round the clock support services including preventative maintenance and instant personal and online access to support analysts.

Rushmore apparently has no compromise on Quality. It has been successful in delivering one hundred percent quality Software Systems. Effective robust Software Quality Assurance Plan (SQAP) and Software Configuration Management Plan (SCMP) are devised for each and every project.

Rushmore's Technology Solution partners program is aimed at delivering the product; training, support and marketing resources to bear that will make each of our partnerships successful. Whether an independent software vendor (ISV), systems integrator (SI), consultant, application service provider (ASP), value-added reseller (VAR), or hardware platform vendor, the Rushmore Solution Partners Program will match your business model and lead to real opportunities.

The strengths of Rushmore include dynamic, knowledgeable employees, emphasis on quality while still delivering on time and complete range of services and capabilities.

1.2 Existing system and its limitations:

Resource Correlator 2 was in use for a limited period of time. Very few users used it. The main disadvantage of the product was that it did not have a Communication module (middle tier) that acts as an interface between the Client and the MediaSeek's database. Hence it required a large amount of Client side

coding and it could not provide software reusability for other MediaSeek's applications.

1.3 Proposed system and its advantages:

Resource Correlator 3 (RCO/3) is an n-tier 32-bit Windows application written with Microsoft Visual C++ 6.0 . The database back-end is MSDE (Microsoft Data Engine). MSDE is 100% scalable to Microsoft's SQL Server 7.0 . Access to the data stored in the MSDE tables is accomplished through the use of ActiveX Data Objects (ADO), which has, in turn, been encapsulated in a middle software tier. This middle data access tier is provided in an ATL-based DLL called Magnolia.

Resource Correlator 3 provides users with the ability to produce:

- Educational *Resource Data* (formerly referred to as *Resource Keys*) correlated to MediaSeek's Knowledge Base for use within MediaSeek's production database and products that utilize this database.
- Correlation Matching Reports that show the matches between correlated educational resources and Guiding Documents.
- Resource Summary Reports that show all the information and correlated knowledge base statements for the components within a given resource.
- Resource-Resource Reports of the matches between components in two resources
- Guiding Document-Guiding Document Reports of the matches between elements in two Guiding Documents

System Requirements

2. *System Requirements*

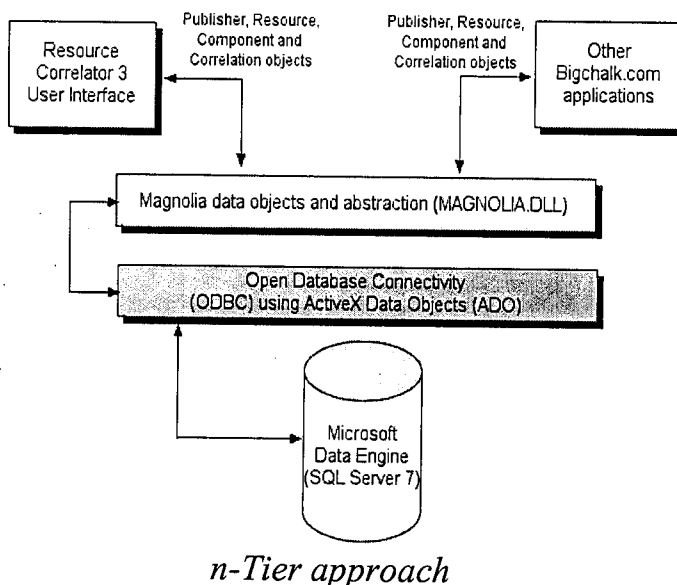
2.1 **Product Definition:**

Mediaseek's Resource Correlator Release 3 is built on what is called an *n-tier methodology* (where *n* is usually 2 or 3). What this means is that different software chores are separated into software components. This provides software reusability for other Mediaseek applications that have yet to be developed or existing applications that are not yet adapted. The middle tier is responsible for interaction between the data source (usually an ODBC-driven database back end such as Microsoft[®] SQL Server 7) and the user interface (UI) that the end-user sees when they are interacting with Resource Correlator.

Why a Middle Tier?

A middle tier insulates the user interface from changes to the database-specific information it must process. What this provides is a way to manage data in a way that is database product non-specific. But doesn't a data access standard such as ODBC or ADO provide this? Yes they do, but for the most primitive types of database chores. Typically ODBC-based coding requires the embedding of standard SQL statements. Data is provided to the calling application in terms of row sets and columns. This places the burden of dealing with both the SQL necessities and the business logic inherent in the application's user interface on the UI developer. This is typically troublesome and error-prone in a development team environment.

Here's the fundamental view of this approach:

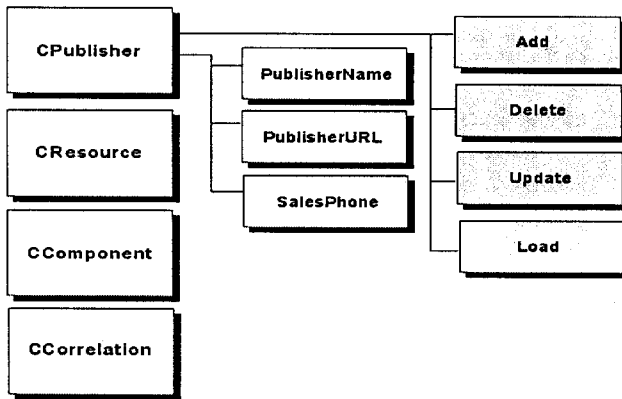


While there's nothing inherently wrong with the application in question using ADO (or a different methodology) for accessing the tables stored in the database, the added benefit in using the Magnolia classes is two-fold: First, there are business rules built into how the data is maintained within Magnolia. Second, a different data engine or ODBC-like layer could be put in place without disturbing the UI codes much at all.

Magnolia Overview:

Magnolia is made up of collections and container objects that adhere to the business rules of Resource Correlations. Each client side database table is wrapped with a COM object. A Collection object is a collection of containers. The container objects expose the actual data while the Collection enables the addition, deletion, and acquisition of container objects. Container objects make up a collection. They contain the actual data in the underlying tables.

Magnolia Classes:



Sample Magnolia classes

When you define an instance of one of these classes, you have an object. With a Publisher object, as an example, you can set its properties or use its methods to perform some sort of action on the object. In the example shown here, those actions are specific to working with the data source.

ATL & COM:

Magnolia is actually implemented using ATL (ActiveX Template Library). In the Windows™ operating systems, this is known as COM, which, in turn, was improved to COM+ with the advent of Windows 2000 technology. COM, or the Component Object Model, is a way for an object to create instances of itself. Magnolia is an ATL version of just such a library. ATL was chosen because it is a lightweight subset of those features most commonly used by third-party controls and applications.

2.2 Product Plan:

Life Cycle Mode:

The Spiral Model is proposed to be in life cycle mode is followed while developing the product software. It provides the potential for the rapid development of incremental version of the software. The software is developed in the series of incremental releases. The spiral model has six tasks region.

Task Region 1:

- Terminology : Customer Communication
- Milestones : Nov 6th – 9th
- Work Product : The project guide introduces the main concepts of the project. A preliminary study is made about Resource Correlator 2 and its limitations. Enhancements to be introduced in the next version are studied.

Task Region 2:

- Terminology : Planning
- Milestones : Nov 10th – 14th
- Work Product : Analysis of the product definition. What function the product has to perform, Processing environment, Product features, Programming language and development tools to be implemented are all decided in this stage.

Task Region 3:

- Terminology : Risk Analysis
- Milestones : Nov 15th – 20th
- Work Product :
- Technical Risk :

Since Design Document plays a very important role in coding, Preparation of Design Documentation consumes more time. Coding is based on all the logics, concepts that can be used. Once the design is made the coding can be started from scratch without any confusion.

Managerial Risk :

For every module time limits are set to be fixed for its completion . The project duration is between Nov 6th – Feb 15th . With in the given time slot for each module it has to be completed. If there is a delay in any module then there is a delay in completion of the whole project. So, the modules have to be completed in the appropriate time.

Task Region 4:

- Terminology : Engineering.
- Milestones : *SRS Document* Nov 25th – 30th
- Work Product : Based on the needs of the project, the software requirement specification is prepared. SRS includes Product Overview, Processing Environment, Functional Specifications , Performance requirements, Design guidelines.

- Milestones : *Design Document* Dec 1st – Dec 14th .
- Work Product : Based on the needs of the project ,the Design document is prepared. Designing plays an important role during coding. Once the design is framed well, the programmer can start the coding very easily. Design document includes Architectural design overview and Detailed design Specification.

Task Region 5:

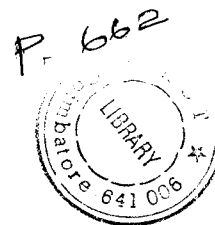
- Terminology : Construction and Release.

Milestones	Work Product
Dec 15 th – 23 rd	Database designing using MS-SQL server 7.0. Creation of all the tables based on the design specified.
Dec 24 th – Jan 9 th	Front end designing using MDI document of Visual C++ 6.0. All the required forms are created as per the specifications in the design document.
Jan 10 th – Jan 16 th	Coding for the Communication module is done. Required header files for the DLL is created using ATL concepts.
Jan 17 th – Jan 28 th	Coding for the Correlation module in which the coding for the features like search & advanced search are also done.

Jan 29 th – Feb 5 th	Collection of Knowledge Base Statements. Registration of the various publishers along with their resources and components is done.
Feb 6 th – Feb 9 th	Preparing Test Plan.
Feb 10 th – Feb 12 th	Installing Exe file.
Feb 14 th	Demonstrating the whole project.

Task Region 6:

- Terminology : Customer Evaluation
- Milestones : Feb 15th – 19th
- Work Product : On seeing the execution , the customer gave the feedback. His feedback was the product works very well and it can be extended in future for other MediaSeek's applications, if time permits.



Software Requirement Specifications

3. *Software Requirements Specification*

3.1 **Hardware Specifications:**

Processor	: Pentium III 450 MHz.
System RAM	: 64 MB.
Cache Memory	: 512 KB.
Floppy Disk Drive	: 1.44 M.B
Hard Disk Drive	: 20 GB.
Keyboard	: Windows Keyboard.
Mouse	: Microsoft Compatible PS/2 Mouse.
Display Adapter	: VGA Card with on-board 8 MB VRAM supporting a screen resolution of 1024*768 with 16 bit color depth running.
Monitor	: Color Monitor with refresh rate of 85 Hz.

3.2 **Software Specifications:**

Operating System	: Windows NT / 98 / 2000
Language Used	: Microsoft Visual C++ 6.0
RDBMS	: Microsoft SQL Server 7.0

Microsoft Visual C++ :

The Microsoft Visual C++ package is provided along with an Integrated Development Environment (IDE). This feature of Visual C++ helps the users in developing complex projects with ease. Developer Studio is used to integrate the development tools and the Visual C++ Compiler. A complex Application can be created by scanning through an impressive amount of online help. The Developer Studio also helps the programmer to debug the Application within this environment itself.

Development Tools:

The Integrated Development Environment includes a number of tools such as:

- An Integrated Editor offers drag & drop of the user interface controls and the syntax highlighting features.
- A Resource Editor is used to create Windows resources such as bitmaps, icons, dialog boxes, and menus.
- An Integrated Debugger enables to run programs and check for any errors. As it is a part of the Developer Studio, it is easy to detect and rectify the bugs.

Developer Studio Wizards:

Visual C++ provides wizards that work in conjunction with the MFC application framework and the Active Template Library (ATL) to create starter programs for the Application Developer. These are collectively known as MFC AppWizard. There are other wizards that help the Application Developer to add functionality to the programs.

- **MFC AppWizard :** It generates a complete suite of source files and resource files based on classes from the MFC library. There are two versions of this wizard, one that helps us create an MFC executable program and one that helps us create an MFC DLL. AppWizard supports three interfaces namely Single Document Interface (SDI), Multiple Document Interface (MDI), and Dialog-based Interface.
- **The ATL COM AppWizard :** It guides the Application Developer through the tasks associated with creating an Active Template Library (ATL) application. Depending on the Wizard options you choose, these applications can be very small and efficient.
- **ClassWizard :** ClassWizard makes it easier to do certain routine tasks such as creating new classes, defining message handlers, overriding virtual functions, and gathering data from controls in a dialog box, form view, or record view. ClassWizard also makes it easy to add properties, methods, and events to automation objects. ClassWizard works with programs that use the Microsoft Foundation Class Library (MFC) or the Active Template Library (ATL).

- **Custom AppWizard :** Using Custom AppWizard, the developers can create their own project type and add it to the list of types available when they create projects. Custom AppWizards are useful for creating generic application project types that can repetitively generate common functionality—application types that can be used over and over again.

MFC Libraries:

The Microsoft Foundation Class (MFC) library offer classes for managing windows objects and offers a number of general purpose classes that can be used in any type of Applications. MFC library represents virtually every Windows API feature and include sophisticated code that streamlines message processing, diagnosing and other details that are normal part of Windows Applications.

MFC makes use of the document/view architecture. MFC supports two types of document/view applications. The first is single-document interface (SDI) applications, which support just one open document at a time. The second is multiple-document interface (MDI) applications, which permit the user to have two or more document open concurrently.

MFC allows you to create full applications, ActiveX controls, and active documents. The MFC library incorporates a robust architecture. It offers automatic message handling and dynamic object typing. It allows self-diagnostics.

Dynamic Link Libraries (DLL):

A dynamic-link library (DLL) is an executable file that acts as a shared library of functions. Dynamic linking provides a way for a process to call a function that is not part of its executable code. The executable code for the function is located in a DLL, which contains one or more functions that are compiled, linked, and stored separately from the processes that use them. DLLs also facilitate the sharing of data and resources.

Multiple applications can simultaneously access the contents of a single copy of a DLL in memory. Many Applications can benefit by being split into a series of main programs and DLLs. There are several linkage options. An MFC library DLL can accommodate entire C++ classes. These DLL-resident classes can be used the same way that statically linked classes are used.

Dynamic linking differs from static linking in that it allows an executable module (either a .DLL or .EXE file) to include only the information needed at run time to locate the executable code for a DLL function. DLLs save memory, reduce swapping, save disk space, upgrade easier, provide after-market support, provide a mechanism to extend the MFC library classes, support multi-language programs, and ease the creation of international versions.

DLLs also facilitate the sharing of data and resources. Multiple applications can simultaneously access the contents of a single copy of a DLL in memory.

ATL and COM:

Component Object Model (COM) is the fundamental "object model" on which ActiveX Controls and OLE is built. COM allows an object to expose its functionality to other components and to host applications. It defines both how the object exposes itself and how this exposure works across processes and across networks. COM also defines the object's life cycle.

ATL is the Active Template Library, a set of template-based C++ classes with which you can easily create small, fast Component Object Model (COM) objects. It has special support for key COM features including: stock implementations, dual interfaces, standard COM enumerator interfaces, connection points, tear-off interfaces, and ActiveX controls.

ATL allows to easily create COM objects, Automation server, and ActiveX controls. ATL provides built-in support for many of the fundamental COM interfaces.

ATL is a fast, easy way to both create a COM component in C++ and maintain a small footprint. ATL can be used to create a control if all of the built-in functionality that MFC automatically provides is not needed. ATL code can be used to create single-threaded objects, apartment-model objects, free-threaded model objects, or both free-threaded and apartment-model objects. ATL is meant to be a lightweight alternative to MFC for creating ActiveX controls.

Database Design:

Database design is one of the most important steps in the system design phase of the system development. A good design of the database can reduce problems like redundancy and anomalies and at the same time enforces integrities like referential integrity, domain integrity, etc. Data Base Management System implements normal databases. The concept called Relational Data Base Management System (RDBMS) is easier to design a database that can enforce all the securities and integrities, which lead to a secured and consistent database.

Data Base Management Concepts:

Databases are normally implemented by Data Base Management System(DBMS). DBMS fall into two broad categories.

- Pointer-driven Systems
- Table-driven Systems

Pointer – driven DBMS use techniques such as participating and training. These are normally host-languages that use high-level language verbs coded within application problems. The designs of database using each system will have a crucial effect on the performance and flexibility of the end result.

Table – driven systems are inverted – file systems that allow the user to set up and maintain database, which may be searched using a wide range of different keys. These systems are generally straightforward to implement.

The user specifies the records and fields, indicated which of the fields will be the keys, and supplies these parameters to the DBMS, which will then set up the database. The effort needed at the design stage is much lesser in this system than pointer – driven systems.

Relational Database Management System (RDBMS):

In RDBMS the data is organized consisting of the rows and columns. There is an explicit pointer stored in the rows. Each table has unique name. To identify a particular row in a table, a column or combination of columns are used. This is called the primary key.

The relational model stores every information in terms of rows and columns of data and thus the relation is automatically established and so relation is implicitly understood. The tabular form of representation forms the basis of implementation of relation in RDBMS.

Microsoft SQL Server 7.0 for Microsoft Windows® 95/98 and Windows NT Workstation is a fully-featured RDBMS targeted for workstation and mobile applications. In addition it is perfect for embedded applications because it provides a fully-featured database engine and core components.

Features of RDBMS - Microsoft SQL Server:

- Dynamic Locking
- Multi-Platform
- Data Warehousing
- Parallel database backup and restore
- Added Memory Support
- Microsoft English Query
- Reliability, Concurrency, Centralized Control

An important step in the database design is the Normalization process. It is the process of breaking up of data and storing them in tables in order to reduce redundancy. Though Normalization reduces redundancy of data, it is desirable to have some degree of redundancy in some cases. In such cases, we deliberately introduce some elements of redundancy for a highly improved performance. This can be regarded as Denormalization.

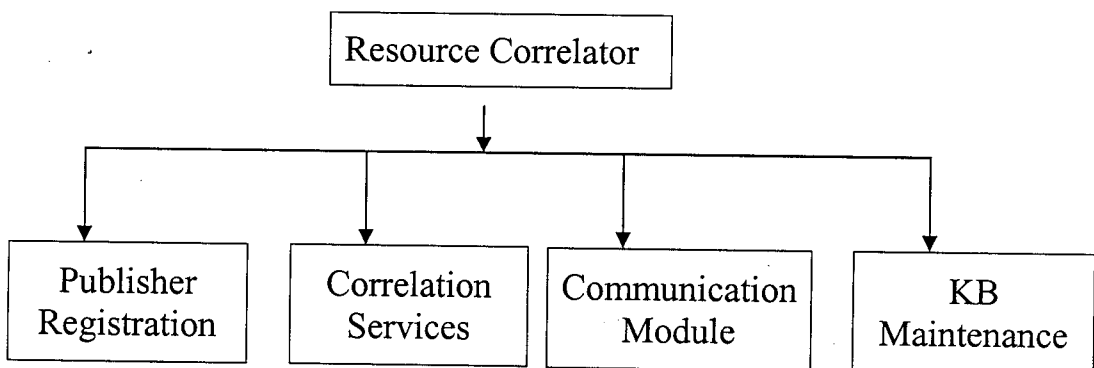
After analyzing all the requirements and inputs, the tables of the system are designed to store the data in a relational manner. Most of the tables in the system are normalized to the extent possible, but a few tables are denormalized in order to reduce the query time and access time.

Indexes are formed on the combination of keys normally used in queries. The Microsoft SQL Server maintains the indexes on primary keys or unique keys and uses implicitly whenever query is fired.

3.3 Functional Specifications:

The 'Resource Correlator' product comprises of four primary modules that are combined to form the whole product. The four main modules that encompass the product are:

- Publisher Registration
- Correlation Services
- Communication Module – Magnolia
- Knowledge Base Maintenance

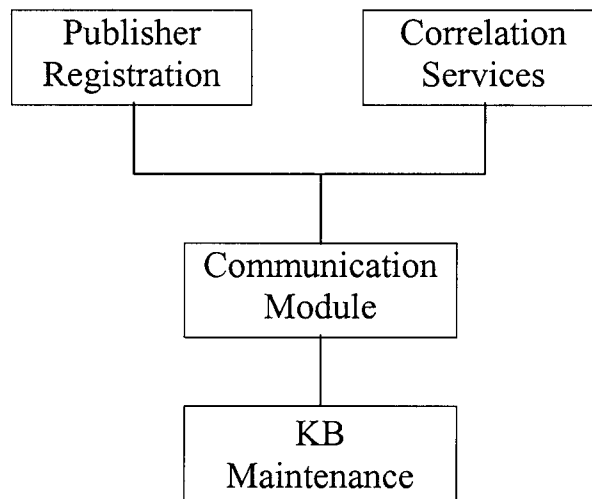


Publisher Registration:

In this module the Publishers or Customers can add or edit the information regarding their Resources and the corresponding Components. Publisher refer to an entity (could be a commercial company, non-profit or independent author) that creates and publishes educational materials. Microsoft Visual C++ MFC library is used to create the user interface to interact with the User.

Correlation Services:

This module enables the user to correlate the components with the Knowledge Base. The user is prompted to select a component and choose the Correlation tab so that appropriate Knowledge Base statements are correlated to the desired Component. It also provides the user with a Search option wherein the user can look for the available Knowledge Base Statements for a desired keyword.



Communication Module:

The Communications module will act as an interface between the Client and the Databases. It comprises the middle-tier development which is called Magnolia . This middle tier insulates the user interface from changes to the database-specific information it must process. This provides software reusability for other applications. It is developed using the ATL and COM technologies.

Knowledge Base Maintenance:

This module is used for managing and maintaining Educational Resource Data that is correlated to MediaSeek's Knowledge Base. The Knowledge Base consists of approximately 12,000 statements that describe what students might learn in the subject areas of English/Language Arts, Mathematics, Science, Social Studies, Technology, and Health/Physical Education. It is built using Microsoft SQL Server.

Design Document

4. Design Document

4.1 Architectural Design Specifications:

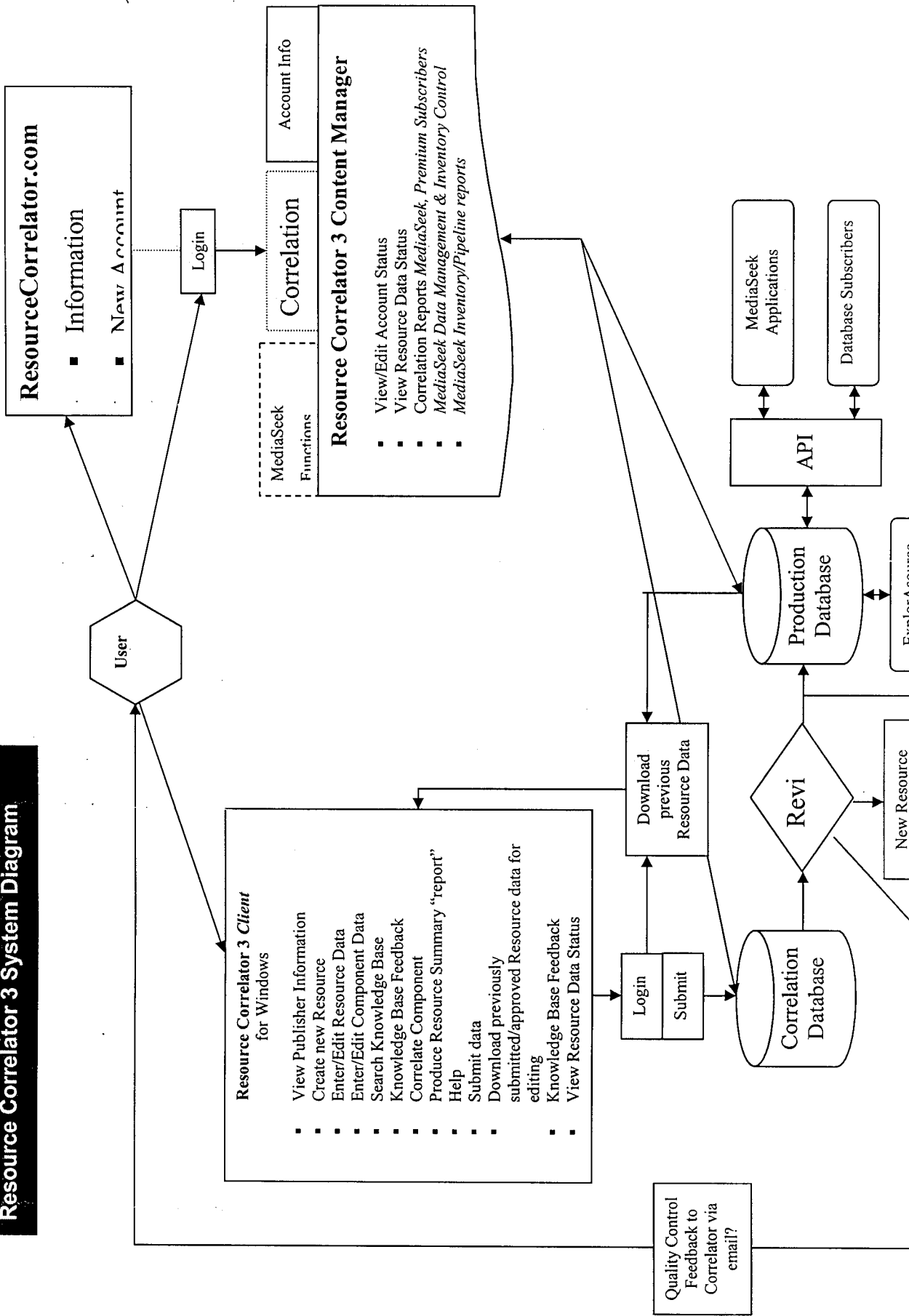
Structure diagram explains the details about the product structure. What are the inputs and outputs to the structure, how the product is comprised into modules and its name is also mentioned.

Resource Correlator 3 consists of five components:

1. An informational web site through which users will be able to download the Client and access user account status
2. Browser-based account status, data and inventory management, and reporting function through the use of Seagate's Info Server and Client, initially available only for internal Mediaseek use.
3. The Resource Correlator 3 Client for Windows NT 4.0/98/2000/ME which has functionality similar to but scaled down from the existing Resource Correlator 2.x Application.
4. A Communications module that will act as an interface between the Client and MediaSeek's databases.
5. A Help and documentation system, including demonstration versions of the Client and browser functionality.

Resource Correlator 3 System Diagram and functional description is represented below:

Resource Correlator 3 System Diagram

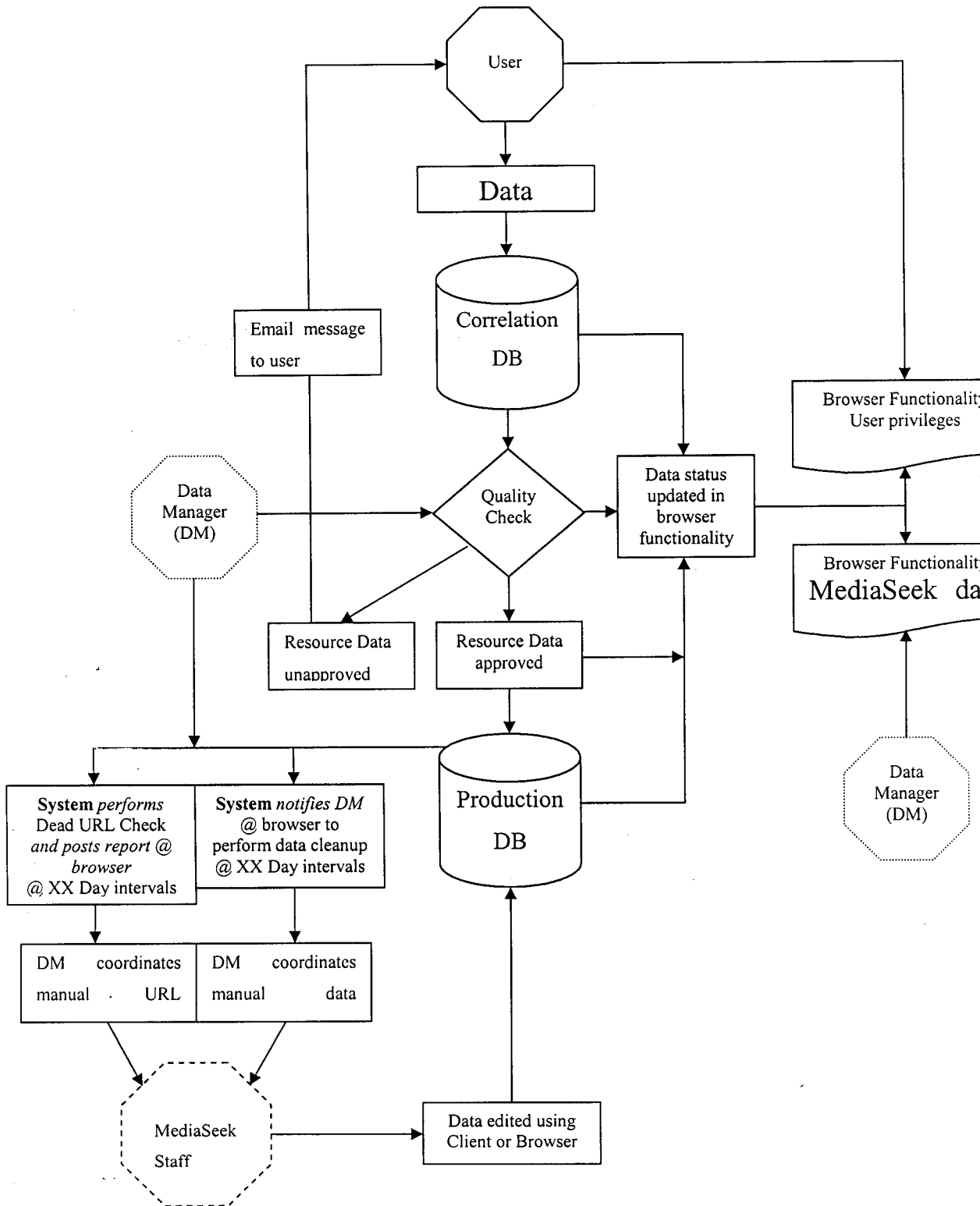


MediaSeek's Data Management Workflow:

This section describes specific features required by MediaSeek Data Manager.

Reference	Action	Location	Result
1.	Receives notification that new Resource Data has been submitted	<i>Via Browser-based MediaSeek functionality TBD</i>	Data Manager notified of submission
2.	Data manager reviews submitted data in Browser-based functionality	DM machine	Resource Data reviewed by human
3.	Data manager accepts data	DM machine	Data flagged as accepted
4.	System flags data as accepted	MediaSeek system; status sent to Client machine during next connection	Data flagged as accepted
5.	Data manager rejects data	MediaSeek system; status sent to Client machine during next connection	Data flagged as unaccepted; notification sent to Client user via email and noted in Account Status
6.	Accepted data passed to MediaSeek Production Database	MediaSeek System/Browser	New data added to Production DB (automatic via db triggers)
7.	Resource Data Cleanup notification	MediaSeek system/Browser	Data manager notified by system to do clean up every XX days
8.	Data manager performs cleanup	MediaSeek system; browser functionality	Updates/edits: Publisher names & contact information Resource names Known inaccuracies
9.	Dead link review	MediaSeek system & WWW	System "automatically" does URL link check every XX days; report generated to Browser functionality; notification sent to Data Manager
10.	Data manager coordinates Dead Link investigation	WWW	Human checks dead links; investigates

Data Management Workflow



Database Structure

PUBLISHER
Publisher_ID
Publisher Name
Street Name
City
Country
Postal Code
Email
URL

REGISTRATION
Registration_ID
Publisher_ID
Account_ID

RESO_LOOK
LID
Description
Display_Order

RESOUR_KEY
RK_ID
Publisher_ID
Name
Copyright_Year

REFER_TYPE
LID
Description
Display_Order

GRADE_LOOK
LID
Description
Display_Order

MEDIA_TYPE
LID
Description
Display_Order

RESOURCE_COMP
RK_ID
CORR_ID
Primary_KB
Name
Description
Reference
Display_Order

AGE_LOOKUP
LID
Description
Display_Order

KBH_LEVEL
LID
Description
Display_Order

KNOWL_BASE
KB_ID
Name
Display_Order

KNOWL_BASE_STAT
KBS_ID
Primary_KB
KBS_TYPE
Statment

GROUP_LOOKUP
LID
Description
Display_Order

CORRELATOR
CORR_ID
User Name
Name
Account_ID

4.2 Table Design:

Table Name : **Publisher**

Description : This table records all Publisher details. Whenever the Publisher registers to the Media Seek Ltd. a record is entered here.

Field Name	Type	Description
Publisher Name	Text	Name of the Publisher
Street Name	Text	Publisher's Street Name
City	Text	Publisher's City
State	Text	Publisher's State
Postal Code	Text	Publisher's Postal Code
Country	Text	Publisher's Country
Customer Phone	Text	Customer's Phone Number
Fax	Text	Publisher's Fax Number
Ordering Phone	Text	Customer Ordering Ph No.
Email Address	Text	Publisher's Email Address
URL	Text	Publisher's URL

Table Name : **Resource Key**

Description : This table has the details of Publisher ID, Resource ID Name, Copyright Year.

Field Name	Type	Description
RK_ID	Number	ID of the Resource Key
Publisher_ID	Number	ID of the Publisher
Name	Text	Name of the Resource
Copyright Year	Text	Copyright Year of the Resource

Table Name : **Knowledge_Base_Statement**

Description : This table has the details of KBS_ID, Primary_KB and various Knowledge Base Statements given by the Correlators.

Field Name	Type	Description
KBS_ID	Number	Reference for KBS
Primary_KB	Number	Reference for Primary KB
Statement	Text	Knowledgebase Statements

Table Name : **Correlator**

Description : This table has the details of the Correlator's ID, User Name and Name of the Correlator and his Account Number.

Field Name	Type	Description
CORR_ID	Text	Reference for the Correlator
User Name	Text	User name of Correlator
Name	Text	Name of the Correlator
Account_ID	Number	Correlator's Account Number

Table Name : **Resource Component**

Description : This table has the details of RK_ID, Corr_ID, Primary_KB, Name of the Resource, Reference.

Field Name	Type	Description
RK_ID	Number	Reference for Resource Key
CORR_ID	Number	Reference for Correlation Key
Primary_KB	Number	Reference for the KB
Name	Text	Name of the Resource
Reference	Text	Reference for the Resource

Table Name : **Knowledge_Base**

Description : This table has the details of KB_ID, Name of the various subjects and the Order in which it has to be displayed.

Field Name	Type	Description
KB_ID	Number	Reference for KB System
Name	Text	Name of various Subjects
Display_Order	Number	Order of subjects displayed

Table Name : **Grade Lookup**

Description : This table has the details of the LID, Description of the Grades, Display order .

Field Name	Type	Description
LID	Number	Reference ID
Description	Text	Various Grades
Display_Order	Number	Order of Grades displayed

Table Name : **Media Type Lookup**

Description : This table has the details of the LID, description of various media and the order in which it has to be displayed.

Field Name	Type	Description
LID	Number	Reference ID
Description	Text	Various Media's
Display_Order	Number	Order of Media's displayed

Table Name : **Age Lookup**

Description : This table has the details of the LID, Descriptions of various age groups and the order in which it has to be displayed.

Field Name	Type	Description
LID	Number	Reference ID
Description	Text	Various Age Group's
Display_Order	Number	Order of Age Group's

Product Testing

5. Product Testing

System Testing:

Testing is done for the various test cases that were developed in order to catch all the problems and exceptions that arise during the real time implementation of the system. The test cases were developed for unit testing as well as module testing.

Unit Testing:

In this testing, individual components are tested to ensure that they operate correctly. Each component is tested independently, without the interference of other system components. With respect to this project, the individual functions in the DLL are treated as components and were tested. Also, the various activities are tested individually.

Module Testing:

A module is a collection of dependent components such as collection of procedures and functions module encapsulates related components so it can be tested with other system components. Each of the activities, Publisher Registration, Correlation Services, Communication Module, Knowledge Base Maintenance. Thus each module is tested in this stage.

Sample tests performed on this product are:

1. Type of test : **Functional Test**

Test Assumption : Correlation selected without component.

Exact Test Stimuli:

If the user chooses the correlation tab without selecting any Component, the Resource Correlator should display a message box to ask the user to select any Component before clicking correlation tab.

Expected Outcome:

The Resource Correlator displays the Message Box to indicate the user that no Component is selected for correlation.

2. Type of test : **Functional Test**

Test Assumption : Adding new entity with any unfilled fields.

Exact Test Stimuli:

If the user chooses to add any new Resource or Component with any fields not entered, the Resource Correlator should display a message box to intimate the user to enter the field to be entered before clicking add button.

Expected Outcome:

The Resource Correlator displays the Message Box to indicate the user to enter the unfilled fields.

Future Enhancements

6. Future Enhancements

The Resource Correlator can be provided online so that the users (Educators, Classroom teachers, Students and Publishers themselves) can register with the Mediaseek to review the Resources and Components pertaining to the desired Publishers. In addition the users must be granted permission to correlate any Component with the available Knowledge Base Statements. Report generation can be included to provide the users to review their Correlation in the future. Additionally, the resources like books, workbooks, videos, CDs and audiotapes can be made purchasable online.

Conclusion

7. Conclusion

The Resource Correlator thus successfully correlated the Components, of the Resources pertaining to any Publisher, to the relevant Knowledge Base Statements. And also the user finds appropriate Knowledge Base Statements for each Component using the additional Search feature.

The Magnolia middle-tier that was developed using the ATL technologies acted as the interface between the client and the databases. As proposed by the functional specifications, it provides software reusability for other applications that have yet to be developed or existing applications that are not yet adapted. Also this product is designed in such a way that it can be enhanced in future with ease.

Bibliography

8. *Bibliography*

- VC++ in 21 Days – Nathan Gurewich & Ori Gurewich
TechMedia Publications - Edition 1996.
- MFC Programming – Richard M Jones
McGraw Hill International - Editions 1998.
- VC++ 6.0 Complete Reference – Chris II Pappas & William H Murray.
Tata McGraw Hill - Edition 1999.
- MSDN Library – <http://www.Microsoft MSDN.com>
- Microsoft SQL Server in 21 Days – Rick Sawtell & Richard Waymire
TechMedia Publications - Edition 1998.
- Microsoft SQL Server – Ronald R Talmage
Prima Publications - Edition 1999.
- ATL Programming in 21 Days – Kenn Scribner
TechMedia Publications.

Annexure

Sample Code

Sample Code

Implementation of Knowledge Base Statement

```
#include "stdafx.h"
#include "Magnolia.h"
#include "KnowledgeBases.h"
#include "KBStatement.h"

STDMETHODIMP CKBStatement::get_knowledgeBase(IDispatch **pVal)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState())
    return CollectionUtils<CKnowledgeBases>::GetRowByID(this, m_parent,
        m_primaryKB, orderByName, pVal);
}

STDMETHODIMP CKBStatement::get_kbsType(BSTR *pVal)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState())
    CString temp( (BSTR)m_kbsType );
    *pVal = temp.AllocSysString();
    return S_OK;
}

STDMETHODIMP CKBStatement::get_indexNum(long *pVal)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState())
    *pVal = m_indexNum;
    return S_OK;
}

STDMETHODIMP CKBStatement::get_statement(BSTR *pVal)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState())
    CString temp( (BSTR)m_statement );
    *pVal = temp.AllocSysString();
    return S_OK;
}

bool CKBStatement::Load()
{
    try
    {
        if(!IContainerImpl<CKBStatements, IKBStatement, &IID_IKBStatement,
            &LIBID_MAGNOLIALib>::Load())
    }
```

```

return false;
ADODB::_Recordset *rset = 0;
if( m_parent->get_recordset((IDispatch **) &rset) != S_OK )
return false;
ADODB::FieldsPtr pFields = rset->GetFields();
ADODB::Field *pFieldDetail;
_variant_t var;
pFieldDetail = pFields->GetItem( "KBS_ID" );
var = pFieldDetail->GetValue();
if( var.vt != VT_NULL )
    m_kbsID = var;
else
    m_kbsID = 0;

pFieldDetail = pFields->GetItem( "KBS_Type" );
var = pFieldDetail->GetValue();
if( var.vt != VT_NULL )
    m_kbsType = var;
else
    m_kbsType = L"";

pFieldDetail = pFields->GetItem( "Index_Num" );
var = pFieldDetail->GetValue();
if( var.vt != VT_NULL )
    m_indexNum = var;
else
    m_indexNum = 0;

pFieldDetail = pFields->GetItem( "Statement" );
var = pFieldDetail->GetValue();
if( var.vt != VT_NULL )
    m_statement = var;
else
    m_statement = L"";
}
catch ( _com_error &e )
{
    RecordError( e.Source(), e.Description() );
    return S_FALSE;
}
return S_OK;
}

```

Implementation of Publisher

```
#include "Magnolia.h"  
#include "CollectionImpl.h"  
#include "Publisher.h"  
#include "ResourceKey.h"  
#include "ResourceKeys.h"  
#include "ResourceManager.h"
```

```
STDMETHODIMP CPublisher::get_publisherTypeLID(long *pVal)  
{  
    AFX_MANAGE_STATE(AfxGetStaticModuleState())  
    *pVal = m_lPublisherTypeLID;  
    return S_OK;  
}
```

```
STDMETHODIMP CPublisher::put_publisherTypeLID(long newVal)  
{  
    AFX_MANAGE_STATE(AfxGetStaticModuleState())  
    m_sPublisherType = newVal;  
    return S_OK;  
}
```

```
STDMETHODIMP CPublisher::get_publisherType(BSTR /**pVal*/)  
{  
    AFX_MANAGE_STATE(AfxGetStaticModuleState())  
    return E_NOTIMPL;  
}
```

```
STDMETHODIMP CPublisher::put_publisherType(BSTR /*newVal*/)  
{  
    AFX_MANAGE_STATE(AfxGetStaticModuleState())  
    return E_NOTIMPL;  
}
```

```
STDMETHODIMP CPublisher::get_publisherName(BSTR *pVal)  
{  
    AFX_MANAGE_STATE(AfxGetStaticModuleState())  
    CString temp( (BSTR)m_sPublisherName );  
    *pVal = temp.AllocSysString();  
    return S_OK;  
}
```

```
STDMETHODIMP CPublisher::put_publisherName(BSTR newVal)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState())
    m_sPublisherName = newVal;
    return S_OK;
}
```

```
STDMETHODIMP CPublisher::get_streetName1(BSTR *pVal)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState())
    CString temp( (BSTR)m_sStreetName1 );
    *pVal = temp.AllocSysString();
    return S_OK;
}
```

```
STDMETHODIMP CPublisher::put_streetName1(BSTR newVal)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState())
    m_sStreetName1 = newVal;
    return S_OK;
}
```

```
STDMETHODIMP CPublisher::get_streetName2(BSTR *pVal)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState())
    CString temp( (BSTR)m_sStreetName2 );
    *pVal = temp.AllocSysString();
    return S_OK;
}
```

```
STDMETHODIMP CPublisher::put_streetName2(BSTR newVal)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState())
    m_sStreetName2 = newVal;
    return S_OK;
}
```

```
STDMETHODIMP CPublisher::get_city(BSTR *pVal)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState())
    CString temp( (BSTR)m_sCity );
    *pVal = temp.AllocSysString();
    return S_OK;
}
```

```

STDMETHODIMP CPublisher::get_resourceKeys(IDispatch **pVal)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState())
    CComObject<CResourceKeys> *resourceKeys;
    if( CComObject<CResourceKeys>::CreateInstance(&resourceKeys) != S_OK )
        return S_FALSE;
    resourceKeys->AddRef();
    ADODB::_Connection *con=0;
    m_parent->get_activeConnection((IDispatch **) &con);
    if( con == 0 )
    {
        ASSERT(FALSE);
        return S_FALSE;
    }

    resourceKeys->Attach(con, this);
    if(m_IPublisherID==0)
    {
        ASSERT(FALSE);
        return S_FALSE;
    }

    if( !resourceKeys->LoadByID(m_IPublisherID, orderBy_name) )
    {
        ASSERT(FALSE);
        delete resourceKeys;
        return S_FALSE;
    }
    *pVal = resourceKeys;
    return S_OK;
}

```

```

bool CPublisher::Load()
{
    IContainerImpl<CPublishers, IPublisher, &IID_IPublisher,
        &LIBID_MAGNOLIALib>::Load();
    if( m_parent == NULL )
        return false;
    ADODB::_Recordset *rset = 0;
    if( m_parent->get_recordset((IDispatch **)&rset) != S_OK )
        return false;
    try
    {

```

```

ADODB::FieldsPtr  pFields = rset->GetFields();
ADODB::Field      *pFieldDetail;

_variant_t        var;
pFieldDetail = pFields->GetItem( "Publisher_ID" );
var = pFieldDetail->GetValue();
if( var.vt != VT_NULL )
    m_lPublisherID = var;
else
    m_lPublisherID = 0;

pFieldDetail = pFields->GetItem("Name");
var = pFieldDetail->GetValue();
if( var.vt != VT_NULL )
    m_sPublisherName = var;
else
    m_sPublisherName = "";

pFieldDetail = pFields->GetItem( "Publisher_Type" );
var = pFieldDetail->GetValue();
if( var.vt != VT_NULL )
    m_sPublisherType = var;
else
    m_sPublisherType = "";

// Street line 1
pFieldDetail = pFields->GetItem( "Street_Name1" );
var = pFieldDetail->GetValue();
if( var.vt != VT_NULL )
    m_sStreetName1 = var;
else
    m_sStreetName1 = "";

// Street line 2
pFieldDetail = pFields->GetItem( "Street_Name2" );
var = pFieldDetail->GetValue();
if( var.vt != VT_NULL )
    m_sStreetName2 = var;
else
    m_sStreetName2 = "";

// City
pFieldDetail = pFields->GetItem( "City" );
var = pFieldDetail->GetValue();
if( var.vt != VT_NULL )
    m_sCity = var;

```



```

else
    m_sCity = "";

// State
pFieldDetail = pFields->GetItem( "State" );
var = pFieldDetail->GetValue();
if( var.vt != VT_NULL )
    m_sState = var;
else
    m_sState = "";

// Zip code
pFieldDetail = pFields->GetItem( "Postal_Code" );
var = pFieldDetail->GetValue();
if( var.vt != VT_NULL )
    m_sPostalCode = var;
else
    m_sPostalCode = "";

// Country
pFieldDetail = pFields->GetItem( "Country" );
var = pFieldDetail->GetValue();
if( var.vt != VT_NULL )
    m_sCountry = var;
else
    m_sCountry = "";

// Customer service number
pFieldDetail = pFields->GetItem( "Cust_Service_Phone" );
var = pFieldDetail->GetValue();
if( var.vt != VT_NULL )
    m_sCustomerService = var;
else
    m_sCustomerService = "";

// EMail address
pFieldDetail = pFields->GetItem( "Email" );
var = pFieldDetail->GetValue();
if( var.vt != VT_NULL )
    m_sEMail = var;
else
    m_sEMail = "";

// Ordering phone number
pFieldDetail = pFields->GetItem( "Ordering_Phone" );
var = pFieldDetail->GetValue();

```

```

if( var.vt != VT_NULL )
    m_sOrderingPhone = var;
else
    m_sOrderingPhone = "";

// Fax number
pFieldDetail = pFields->GetItem( "Fax" );
var = pFieldDetail->GetValue();
if( var.vt != VT_NULL )
    m_sFax = var;
else
    m_sFax = "";

// Address indicator (whatever that is...)
pFieldDetail = pFields->GetItem( "Address_Indicator" );
var = pFieldDetail->GetValue();
if( var.vt != VT_NULL )
    m_sAddressIndicator = var;
else
    m_sAddressIndicator = "";

// Time/date stamp
pFieldDetail = pFields->GetItem( "DateStamp" );
var = pFieldDetail->GetValue();
if( var.vt != VT_NULL )
    m_dtDateStamp = var;
else
    m_dtDateStamp = 0;

// Publisher's web page address (URL)
pFieldDetail = pFields->GetItem( "URL" );
var = pFieldDetail->GetValue();
if( var.vt != VT_NULL )
    m_sURL = var;
else
    m_sURL = "";
}
catch ( _com_error &e)
{
    RecordError( e.Source(), e.Description() );

    return false;
}

return true;

```

```
}
```

```
STDMETHODIMP CPublisher::get_fax(BSTR *pVal)
```

```
{
```

```
    AFX_MANAGE_STATE(AfxGetStaticModuleState())
```

```
    CString temp( (BSTR)m_sFax );
```

```
    *pVal = temp.AllocSysString();
```

```
    return S_OK;
```

```
}
```

```
STDMETHODIMP CPublisher::put_fax(BSTR newVal)
```

```
{
```

```
    AFX_MANAGE_STATE(AfxGetStaticModuleState())
```

```
    m_sFax = newVal;
```

```
    return S_OK;
```

```
}
```

```
bool CPublisher::Export( CResourceManager &rm )
```

```
{
```

```
    #define CHECK_WRITE( key, value ) \
```

```
    if( !rm.WriteData( key, value ) ) \
```

```
        return false;
```

```
    rm.SetSectionName( "Entities" );
```

```
    CHECK_WRITE( "PUBLISHER", "1" );
```

```
    rm.SetSectionName( "PUBLISHER - 1" );
```

```
    CHECK_WRITE( "PUBLISHER_ID", m_lPublisherID );
```

```
    CHECK_WRITE( "PUBLISHER_TYPE", m_sPublisherType );
```

```
    CHECK_WRITE( "NAME", m_sPublisherName );
```

```
    CHECK_WRITE( "STREET_NAME1", m_sStreetName1 );
```

```
    CHECK_WRITE( "STREET_NAME2", m_sStreetName2 );
```

```
    CHECK_WRITE( "CITY", m_sCity );
```

```
    CHECK_WRITE( "STATE", m_sState );
```

```
    CHECK_WRITE( "POSTAL_CODE", m_sPostalCode );
```

```
    CHECK_WRITE( "COUNTRY", m_sCountry );
```

```
    CHECK_WRITE( "CUST_SERVICE_PHONE", m_sCustomerService );
```

```
    CHECK_WRITE( "FAX", m_sFax );
```

```
    CHECK_WRITE( "EMAIL", m_sEMail );
```

```
    CHECK_WRITE( "ORDERING_PHONE", m_sOrderingPhone );
```

```
    CHECK_WRITE( "ADDRESS_INDICATOR", m_sAddressIndicator );
```

```
    CHECK_WRITE( "DATESTAMP", m_dtDateStamp );
```

```
    CHECK_WRITE( "URL", m_sURL );
```

```
    return true;
```

```
}
```

Implementation of Correlator

```
#include "stdafx.h"  
#include "Magnolia.h"  
#include "Correlator.h"
```

```
STDMETHODIMP CCorrelator::get_userName(BSTR *pVal)  
{  
    AFX_MANAGE_STATE(AfxGetStaticModuleState())  
    CString temp( (BSTR)m_sUserName );  
    *pVal = temp.AllocSysString();  
    return S_OK;  
}
```

```
STDMETHODIMP CCorrelator::get_name(BSTR *pVal)  
{  
    AFX_MANAGE_STATE(AfxGetStaticModuleState())  
    CString temp( (BSTR)m_sName );  
    return S_OK;  
}
```

```
STDMETHODIMP CCorrelator::get_active(BOOL *pVal)  
{  
    AFX_MANAGE_STATE(AfxGetStaticModuleState())  
    *pVal = m_bActive;  
    return S_OK;  
}
```

```
STDMETHODIMP CCorrelator::get_accountID(long *pVal)  
{  
    AFX_MANAGE_STATE(AfxGetStaticModuleState())  
    *pVal = m_lAccountID;  
    return S_OK;  
}
```

```
bool CCorrelator::Load()  
{  
    try  
    {  
        if(!IContainerImpl<CCorrelators, ICorrelator, &IID_ICorrelator,  
            &LIBID_MAGNOLIALib>::Load())  
            return false;  
        ADODB::_Recordset *rset = 0;  
        if( m_parent->get_recordset((IDispatch **)&rset) != S_OK )  
            return false;  
    }  
}
```

```

ADODB::FieldsPtr  pFields = rset->GetFields();
ADODB::Field      *pFieldDetail;

_variant_t  var;
pFieldDetail = pFields->GetItem( "Corr_ID" );
var = pFieldDetail->GetValue();
if( var.vt != VT_NULL )
    m_corrID = var;
else
    m_corrID = 0;

pFieldDetail = pFields->GetItem( "UserName" );
var = pFieldDetail->GetValue();
if( var.vt != VT_NULL )
    m_sUserName = var;
else
    m_sUserName = L"";

pFieldDetail = pFields->GetItem( "Name" );
var = pFieldDetail->GetValue();
if( var.vt != VT_NULL )
    m_sName = var;
else
    m_sName = "";

pFieldDetail = pFields->GetItem( "Active" );
var = pFieldDetail->GetValue();
if( var.vt != VT_NULL )
    m_bActive = (bool)var;
else
    m_bActive = 0;

pFieldDetail = pFields->GetItem( "Account_ID" );
var = pFieldDetail->GetValue();
if( var.vt != VT_NULL )
    m_lAccountID = var;
else
    m_lAccountID = 0;
}
catch (_com_error &e)
{
    RecordError( e.Source(), e.Description() );
    return S_FALSE;
}
return S_OK;
}

```

Implementation of Magnolia

```
#include "stdafx.h"
#include "Magnolia.h"
#include "Publishers.h"
#include "ResourceKeys.h"
#include "ReferenceTypes.h"
#include "TargetUsers.h"
#include "ages.h"
#include "Grades.h"
#include "Groupings.h"
#include "MediaTypes.h"
#include "ResourceComponents.h"
#include "Correlators.h"
#include "MagnoliaDS.h"
#include "KnowledgeBases.h"

long CMagnoliaDS::m_newID=0;

STDMETHODIMP CMagnoliaDS::Open(BSTR server, BSTR database, BSTR
userName, BSTR password)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState())

    if( m_connection != NULL )
    {
        m_connection.Release();
        m_connection = NULL;
    }

    m_server = server;
    m_database = database;
    m_userName = userName;
    m_password = password;

    if( m_server.length() == 0 )
        m_server = _T("(local)");

    if( m_database.length() == 0 )
        m_database = _T("MediaSeek");

    bool useDefaultPW = true;
    if( m_userName.length() == 0 )
```

```

        m_userName = _T("TabRCO");
else
    useDefaultPW = false;

if( m_password.length() == 0 && useDefaultPW )
    m_password = _T("!@#AfGAn*.1");

CString      s;
s.Format(IDS_CONNECTIONSTRING, (LPCTSTR) m_server, (LPCTSTR)
m_database, (LPCTSTR) m_userName, (LPCTSTR) m_password );
_bstr_t sConnectionString(s);

try
{
    HRESULT hr=0;
        TESTHR(m_connection.CreateInstance(
__uuidof(ADODB::Connection));
        TESTHR(m_connection->Open( sConnectionString, BSTR(""),
BSTR(""), NULL ));
        m_bOpen = true;
        return S_OK;
}
catch(_com_error &e)
{
    _bstr_t sErrorSource( e.Source());
    _bstr_t sErrorDesc( e.Description());

    m_sErrorSource          = sErrorSource;
    m_sErrorDescription = sErrorDesc;
    return S_FALSE;
}
}

```

```

STDMETHODIMP CMagnoliaDS::get_publishers(IDispatch **pVal)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState())
    CComObject<CPublishers> *publishers=0;

    if( CComObject<CPublishers>::CreateInstance(&publishers) != S_OK)
        return S_FALSE;
    publishers->AddRef();
    publishers->Attach(m_connection);
    *pVal = publishers;
    return publishers->LoadAll(orderByName);
}

```

```
STDMETHODIMP CMagnoliaDS::GetResourcesByStatus(long statusID, IDispatch  
**rkeys)
```

```
{  
    AFX_MANAGE_STATE(AfxGetStaticModuleState())  
  
    CComObject<CResourceKeys> *resources;  
  
    if( CComObject<CResourceKeys>::CreateInstance(&resources) != S_OK)  
        return S_FALSE;  
  
    resources->Attach(m_connection);  
    *rkeys = resources;  
  
    return resources->LoadByStatus(statusID, orderByName);  
}
```

```
STDMETHODIMP CMagnoliaDS::get_availReferenceTypes(IDispatch **pVal)
```

```
{  
    AFX_MANAGE_STATE(AfxGetStaticModuleState())  
  
    if(!CollectionUtils<CReferenceTypes>::GetAvail(m_connection, pVal) )  
        return S_FALSE;  
  
    return S_OK;  
}
```

```
STDMETHODIMP CMagnoliaDS::get_availKnowledgeBases(IDispatch **pVal)
```

```
{  
    AFX_MANAGE_STATE(AfxGetStaticModuleState())  
  
    if(!CollectionUtils<CKnowledgeBases>::GetAvail(m_connection, pVal) )  
        return S_FALSE;  
  
    return S_OK;  
}
```

```
STDMETHODIMP CMagnoliaDS::get_availTargetUsers(IDispatch **pVal)
```

```
{  
    AFX_MANAGE_STATE(AfxGetStaticModuleState())  
  
    if(!CollectionUtils<CTargetUsers>::GetAvail(m_connection, pVal) )  
        return S_FALSE;  
  
    return S_OK;  
}
```



```

STDMETHODIMP CMagnoliaDS::get_availStatuses(IDispatch **pVal)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState())

    if(!CollectionUtils<CStatuses>::GetAvail(m_connection, pVal) )
        return S_FALSE;

    return S_OK;
}

STDMETHODIMP CMagnoliaDS::get_availAges(IDispatch **pVal)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState())

    if(!CollectionUtils<CAges>::GetAvail(m_connection, pVal) )
        return S_FALSE;

    return S_OK;
}

STDMETHODIMP CMagnoliaDS::get_availGrades(IDispatch **pVal)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState())

    if(!CollectionUtils<CGrades>::GetAvail(m_connection, pVal) )
        return S_FALSE;

    return S_OK;
}

STDMETHODIMP CMagnoliaDS::get_availGroupings(IDispatch **pVal)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState())

    if(!CollectionUtils<CGroupings>::GetAvail(m_connection, pVal) )
        return S_FALSE;

    return S_OK;
}

STDMETHODIMP CMagnoliaDS::get_availTeachingStrategies(IDispatch **pVal)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState())

    if(!CollectionUtils<CTeachingStrategies>::GetAvail(m_connection, pVal) )
        return S_FALSE;
}

```

```

        return S_OK;
    }

STDMETHODIMP CMagnoliaDS::get_availMediaTypes(IDispatch **pVal)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState())

    if(!CollectionUtils<CMediaTypes>::GetAvail(m_connection, pVal) )
        return S_FALSE;

    return S_OK;
}

STDMETHODIMP CMagnoliaDS::get_availResourceTypes(IDispatch **pVal)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState())

    if( !CollectionUtils<CResourceTypes>::GetAvail(m_connection, pVal))
        return S_FALSE;

    return S_OK;
}

STDMETHODIMP CMagnoliaDS::BeginTransaction()
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState())
    return m_connection->BeginTrans();
}

STDMETHODIMP CMagnoliaDS::AbortTrans()
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState())
    return m_connection->RollbackTrans();
}

STDMETHODIMP CMagnoliaDS::CommitTrans()
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState())
    return m_connection->CommitTrans();
}

STDMETHODIMP CMagnoliaDS::GetNewResourceKey(IDispatch **pVal)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState())
    long rkid = GetNextId( TEXT( "NEXT_RK_ID" ) );
    if( rkid == -1 )

```

```

        return S_FALSE;
    CComObject<CResourceKey> *ikey;
    if( CComObject<CResourceKey>::CreateInstance(&ikey) != S_OK )
        return S_FALSE;
    CResourceKey *rkey = dynamic_cast<CResourceKey *>(ikey);
    rkey->SetRKID(rkid);
    ikey->AddRef();
    *pVal = ikey;
    return S_OK;
}

```

```

STDMETHODIMP CMagnoliaDS::GetNewResourceComponent(IDispatch **pVal)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState())
    long rcid = GetNextId( TEXT( "NEXT_RC_ID" ) );
    if( rcid == -1 )
        return S_FALSE;
    CComObject<CResourceComponent> *icomp;
    if( CComObject<CResourceComponent>::CreateInstance(&icomp) != S_OK )
        return S_FALSE;
    CResourceComponent *rcomp = dynamic_cast<CResourceComponent
*>(icomp);
    if( rcomp == 0 )
        return S_FALSE;
    rcomp->SetRCID(rcid);
    icomp->AddRef();
    *pVal = icomp;
    return S_OK;
}

```

```

STDMETHODIMP CMagnoliaDS::get_availCorrelators(IDispatch **pVal)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState())

    if(!CollectionUtils<CCorrelators>::GetAvail(m_connection, pVal, orderByName)
)
        return S_FALSE;
    return S_OK;
}

```

```

STDMETHODIMP CMagnoliaDS::get_availKBStatements(IDispatch **pVal)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState())
    if(!CollectionUtils<CKBStatements>::GetAvail(m_connection, pVal,
orderByStatement) )
        return S_FALSE;
}

```

```

        return S_OK;
    }

STDMETHODIMP CMagnoliaDS::get_availCorrGrades(IDispatch **pVal)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState())

    if(!CollectionUtils<CRC_CorrByGrades>::GetAvail(m_connection, pVal,
        orderByDescription) )
        return S_FALSE;
    return S_OK;
}

bool CMagnoliaDS::UpdateNextId( LPCTSTR pColumn, long pNewValue )
{
    CString sqlStmt;
    sqlStmt.Format( "UPDATE REGISTRATION SET %s = %d", pColumn,
        pNewValue );
    ADODB::_RecordsetPtr rset;
    if( rset.CreateInstance( __uuidof( ADODB::Recordset ) ) != S_OK )
        return false;
    HRESULT hr = rset->Open(_bstr_t(sqlStmt), (IDispatch *)
        m_connection.GetInterfacePtr(), ADODB::adOpenStatic,
        ADODB::adLockOptimistic, ADODB::adCmdText );
    return SUCCEEDED( hr );
}

long CMagnoliaDS::GetNextId( LPCTSTR pColumn )
{
    CString sqlStmt;
    sqlStmt.Format( "SELECT REGISTRATION_ID, STATIC_MULTIPLIER, %s
        FROM REGISTRATION", pColumn );
    ADODB::_RecordsetPtr rset;
    if( rset.CreateInstance( __uuidof( ADODB::Recordset ) ) != S_OK )
        return -1;
    if( rset->Open(_bstr_t(sqlStmt), (IDispatch *) m_connection.GetInterfacePtr(),
        ADODB::adOpenStatic, ADODB::adLockOptimistic, ADODB::adCmdText ) !=
        S_OK )
        return -1;
    long regId = -1, staticMultiplier = -1, nextId = -1;
    while( !rset->EndOfFile ) {
        ADODB::_FieldsPtr    pFields = rset->GetFields();
        ADODB::_Field        *pFieldDetail;
        _variant_t var;
        pFieldDetail = pFields->GetItem( "REGISTRATION_ID" );
        var = pFieldDetail->GetValue();
    }
}

```

```
if( var.vt != VT_NULL )
    regId = var;
```

```
pFieldDetail = pFields->GetItem( "STATIC_MULTIPLIER" );
var = pFieldDetail->GetValue();
if( var.vt != VT_NULL )
    staticMultiplier = var;
```

```
pFieldDetail = pFields->GetItem( pColumn );
var = pFieldDetail->GetValue();
if( var.vt != VT_NULL )
    nextId = var;
break;
```

```
}
```

```
if( regId == -1 || staticMultiplier == -1 || nextId == -1 )
    return -1;
```

```
long baseValue = regId * staticMultiplier + 1;
if( nextId < baseValue )
    nextId = baseValue;
```

```
long retId = nextId;
nextId++;
if( !UpdateNextId( pColumn, nextId ) )
    return -1;
return retId;
```

```
}
```

```
STDMETHODIMP CMagnoliaDS::updateToggle(long pDependency, long pId, long
pIndexnum, BSTR pName, long *pRetVal)
```

```
{
```

```
AFX_MANAGE_STATE(AfxGetStaticModuleState())
CString tempinput;
tempinput.Format("%d-%d-%d-%s", pDependency, pId, pIndexnum, pName);
CString temp = TEXT(pName);
tempinput.Format("%s", temp);
*pRetVal = UpdateDep( pDependency, pId, pIndexnum, pName);
return S_OK;
```

```
}
```

```
long CMagnoliaDS::UpdateDep( long pDependency, long pId, long pIndexnum, BSTR
pName)
```

```
{
```

```
    CString getsqlStmt;
```

```

CString name = TEXT(pName);
getsqlStmt.Format( "SELECT KB_ID FROM KNOWLEDGE_BASE WHERE
NAME LIKE '%s'", name );
ADODB::_RecordsetPtr getrset;
if( getrset.CreateInstance( __uuidof( ADODB::Recordset ) ) != S_OK )
    return 0;
if( getrset->Open(_bstr_t(getsqlStmt), (IDispatch *)
m_connection.GetInterfacePtr(), ADODB::adOpenStatic,
ADODB::adLockOptimistic, ADODB::adCmdText ) != S_OK )
    return 0;
long primary_kb = -1;

while( !getrset->EndOfFile ) {
    ADODB::FieldsPtr pFields = getrset->GetFields();
    ADODB::Field *pFieldDetail;
    _variant_t var;
    pFieldDetail = pFields->GetItem( "KB_ID" );
    var = pFieldDetail->GetValue();
    if( var.vt != VT_NULL )
        primary_kb = var;

    break;
}
CString tempprimary_kb;
tempprimary_kb.Format("%d", primary_kb);
CString sqlStmt;

sqlStmt.Format( "SELECT KBS_ID FROM
KNOWLEDGE_BASE_STATEMENT WHERE INDEX_NUM = %d AND
PRIMARY_KB = %d", pIndexnum, primary_kb );

ADODB::_RecordsetPtr rset;
if( rset.CreateInstance( __uuidof( ADODB::Recordset ) ) != S_OK )
    return 0;

if( rset->Open(_bstr_t(sqlStmt), (IDispatch *) m_connection.GetInterfacePtr(),
ADODB::adOpenStatic, ADODB::adLockOptimistic, ADODB::adCmdText ) !=
S_OK )
    return 0;

long kbs_id = -1;

while( !rset->EndOfFile ) {
    ADODB::FieldsPtr pFields = rset->GetFields();
    ADODB::Field *pFieldDetail;
    _variant_t var;

```

```

        pFieldDetail = pFields->GetItem( "KBS_ID" );
        var = pFieldDetail->GetValue();
        if( var.vt != VT_NULL )
            kbs_id = var;

        break;
    }

    CString tempkbsid;
    tempkbsid.Format ("%d", kbs_id);
    CString updateStmt;
    updateStmt.Format( "UPDATE RC_CORR_GRADE_LINK SET
    DEPENDENCY = %d WHERE ID = %d AND KBS_ID = %d", pDependency,
    pId, kbs_id );

    ADODB::_RecordsetPtr rsetUpdate;
    if( rsetUpdate.CreateInstance( __uuidof( ADODB::Recordset ) ) != S_OK )
        return 0;

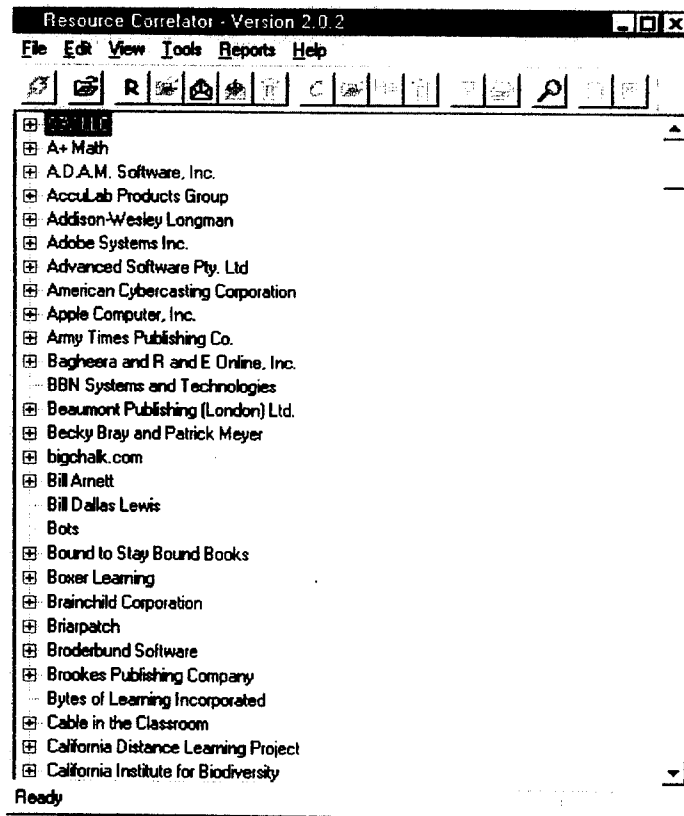
    HRESULT hr = rsetUpdate->Open(_bstr_t(updateStmt), (IDispatch *)
    m_connection.GetInterfacePtr(), ADODB::adOpenStatic,
    ADODB::adLockOptimistic, ADODB::adCmdText );

    if(SUCCEEDED( hr ))
        return 1;
    else
        return -1;
}

```

User Interface

Publisher Tree List



Publisher Screen

Resource Calculator Client 3.0

File Edit View Tools Help

Component Count: Field Operator Value(s)

Publisher Resources Components Correlations

Name:

Street:

Street 2:

City:

State:

Zip:

Country:

Customer Phone:

Ordering Phone:

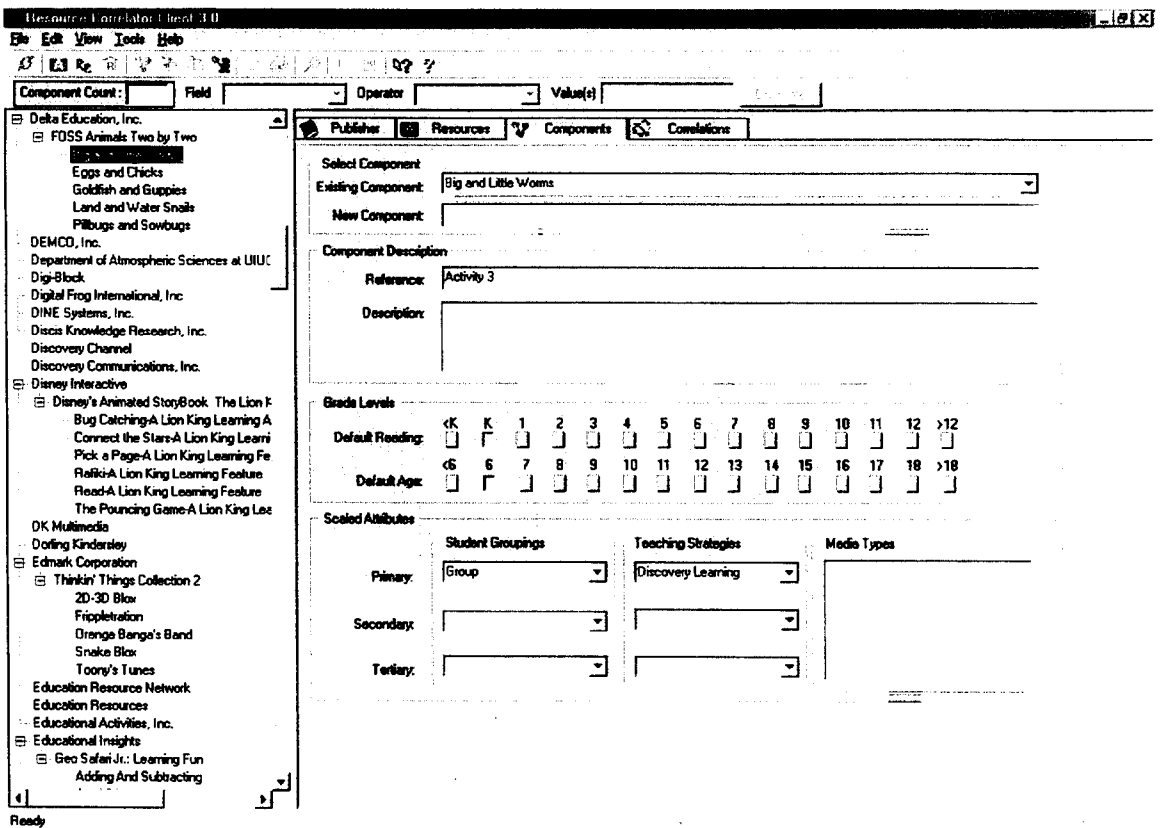
E-Mail Address:

URL/Web Page:

Ready

- FDSS Animals Two by Two
 - Big and Little Worms
 - Eggs and Chicks
 - Goldfish and Guppies
 - Land and Water Snails
 - Pillbugs and Sowbugs
- DEMCO, Inc.
 - Department of Atmospheric Sciences at UIUC
 - Digi-Block
 - Digital Frog International, Inc.
 - DINE Systems, Inc.
 - Discs Knowledge Research, Inc.
 - Discovery Channel
 - Discovery Communications, Inc.
- Disney Interactive
 - Disney's Animated StoryBook: The Lion F
 - Bug Catching-A Lion King Learning A
 - Connect the Stars-A Lion King Learni
 - Pick a Page-A Lion King Learning Fe
 - Refik-A Lion King Learning Feature
 - Read-A Lion King Learning Feature
 - The Pouncing Game-A Lion King Lee
- OK Multimedia
 - Dotting Kinderley
- Edmark Corporation
 - Thinkin' Things Collection 2
 - 2D-3D Block
 - Fippelation
 - Orange Bangs's Band
 - Snake Block
 - Toony's Tunes
- Education Resource Network
 - Education Resources
 - Educational Activities, Inc.
- Educational Insights
 - Geo Safari Jr.: Learning Fun
 - Adding And Subtracting

Components Screen



Correlations Screen

Resource Locator Library 4.0

File Edit View Tools Help

Component Count: Field Operator Value(s)

Publishers Resources Components Correlations

Knowledge Base Statements

Resource: FOSS Animals Two by Two

Component: Big and Little Worms

KBS	ID	Statement	Type

Expanded Text

Completed Statements

KBS	ID	Statement	M1..	M2..	Save List
Science	2057	Animals need food, air, water, and a place to live.	K1..	K1..	<input type="button" value="Save List"/>
Science	1947	Living things can respond to stimuli.	K1..	K1..	<input type="button" value="Save List"/>
Science	1348	Although there is great diversity in living things, there is also great similarity.	K1..	K1..	<input type="button" value="Save List"/>
Science	1146	An animal's body structure is related to how it survives in its environment.	K1..	K1..	<input type="button" value="Save List"/>
Science	1132	Handling and caring for living things.	K1..	K1..	<input type="button" value="Save List"/>
Science	1075	Observing	K1..	K1..	<input type="button" value="Save List"/>
Science	727	Comparing the similarities and differences in behavior among various groups.	K1..	K1..	<input type="button" value="Save List"/>
Science	43	Animals vary in size, body coverings, and ways of moving.	K1..	K1..	<input type="button" value="Save List"/>

4 3

Ready

Advanced Search Screen

The image shows a screenshot of a software interface titled "Search Knowledge Base(s)". The interface is dark with several white input fields and buttons. At the top, there is a title bar with the text "Search Knowledge Base(s)". Below the title bar, there is a large, empty white rectangular area. In the lower section, there are several input fields and buttons. The first field is labeled "statement" and contains the text "statement". To its right is a field labeled "is" containing the text "is". Further right is another empty field. Below these fields, there are several buttons, including one labeled "F" and another labeled "F".