HIDE - METADATA BASED DATA INTEGRATION ENVIRONMENT FOR HYDROLOGICAL DATASETS

Nimmy Ravindran

Thesis submitted to the faculty of the Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
In
Electrical and Computer Engineering

Dec 1st, 2004
Alexandria Research Institute,
Alexandria, Virginia

Keywords: Scientific data integration, data interoperability, metadata, distributed information systems, earth science data.

HIDE - METADATA BASED DATA INTEGRATION ENVIRONMENT FOR HYDROLOGICAL DATASETS

Nimmy Ravindran

ABSTRACT

Efficient data integration is one of the most challenging problems in data management, interoperation and analysis. The Earth science data which are heterogeneous are collected at various geographical locations for scientific studies and operational uses. The intrinsic problem of archiving, distributing and searching such huge scientific datasets is compounded by the heterogeneity of data and queries, thus limiting scientific analysis, and generation/validation of hydrologic forecast models. The data models of hydrologic research communities such as National Weather Service (NWS), National Oceanic and Atmospheric Administration (NOAA), and US Geological Survey (USGS) are diverse and complex. A complete derivation of any useful hydrological models from data integrated from all these sources is often a time consuming process.

One of the current trends of data harvesting in scientific community is towards a distributed digital library initiative. However, these approaches may not be adequate for data sources / entities who do not want to "upload" the data into a "data pool." In view of this, we present here an effective architecture to address the issues of data integration in such a diverse environment for hydrological studies. The heterogeneities in these datasets are addressed based on the autonomy of data source in terms of design, communication, association and execution using a hierarchical integration model. A metadata model is also developed for defining data as well as the data sources, thus providing a uniform view of the data for different kind of users. An implementation of the model using web based system that integrates widely varied hydrology datasets from various data sources is also being developed.

Acknowledgements

I would like to thank my research and academic advisor Dr. Yao Liang for his guidance and support during this program. His advice, patience and optimism have helped me tremendously in the realization of this work. This has been an amazing learning experience. I would also like to thank Dr. Lamine Mili and Dr. Luiz daSilva for reviewing this work and providing valuable suggestions and comments.

I wish to express my gratitude to the National Weather Service River Forecast Center for their financial support in developing the architecture and implementation of the models. Special Thanks go to Dr Xu Liang, Department of Civil and Environmental Engineering, University of California, for reviewing my work and providing valuable insights and comments. Thanks also to Thomas Adams, NWS Ohio River Forecast Center for his patience in reviewing and testing the implementation of the models.

My deepest gratitude goes to the faculty of ECE department and all the staff at Alexandria Research Institute. It has been an honor to work with you in these past years. Very special Thanks to Dr. Binoy Ravindran, for his invaluable attention, and guidance throughout this program. Thank you for your encouragement, and directions as a professor, advisor and family. I will be in eternal debt to you.

This thesis is dedicated to my family and my husband. I owe it to my parents for their love and support in this exciting experience. You have taught me to fight hard, face the adversities and be happy. Mahesh, without you standing beside me, and guiding me through every difficult time in my life, this dream would not have been possible. You are my inspiration.

Table of Contents

Table of Figures

Table of Tables

Chapter 1 . INTRODUCTION

1.1. Motivation

Water is a vital resource for both human needs and natural ecosystems. The tremendous increase in population (e.g., 6 billion people in the 21st century) places a severe burden on the natural environment. Today, about one-third of world's population lives in countries that are experiencing moderate to high water stress (UN-SWI, 1997). On the other hand, severe weather causes deadly flooding hazards that are socially and economically devastating. Therefore, investigations, for example, on trends and spatial distribution of precipitation over the global, and on improvement of hydrologic model forecasts of future floods and available water play an invaluable role in hazard mitigation (e.g., droughts, and floods), agriculture and food production, human health, municipal and industrial supply, environmental quality, and sustainable development in the changing global environment. However, more accurate study on the trends and spatial distribution of precipitation and more accurate hydrologic forecasts rely on better and quicker access, share, and analysis of heterogeneous datasets, measured and transmitted by various technologies, and on better data visualizations.

With the advent of global observing systems (e.g., satellite remote sensing) and global field programs, unprecedented large amounts of critical multi-variate data have been or are being generated. These new data offer great potentials for researchers to discover new hydrologic findings/relationships and to make better forecasts if the data are made accessible and retrievable promptly and appropriately for hydrologic studies. However, these large-scale heterogeneous datasets are presented, over time, with very different structures and formats that require substantial efforts to process in order to be able to make any effective use of them, even with a number of current available software tools, due to rapid advancement of technology to measure and transmit them. Thus, how to use these data promptly and effectively to improve hydrologic research and operational forecast skills poses great challenges. In fact, data preparation generally becomes one of the most time-consuming, expensive, and tedious tasks, and sometimes could take as much as 80% of the overall effort (Anand and Buchner, 1998). Clearly, if such challenges and issues in data acquisition, processing, and management are not sufficiently addressed, significant improvement in hydrologic research would not be achieved efficiently.

1.2. Data Management Methods in Scientific Community

The burden of accessing and processing the heterogeneous data is worse for the hydrologic community compared to others such as oceanic, atmospheric, and ecological areas. In the oceanic area, for example, DODS (Distributed Ocean Data System) provides tools (DODS servers) to simplify scientific data networking issues, and for transforming existing applications into DODS clients (i.e., enabling them to remotely access DODS served data).. In the atmospheric community, the Unidata Program Center (UPC, 1998) under University Corporation for Atmospheric Research (UCAR) offers software and services that enable universities to acquire and use atmospheric and related data on their computers. In the ecological area, a system called Science Environment for Ecological Knowledge (SEEK), is under development. SEEK is an information technology framework and infrastructure that will be used to derive and extend ecological knowledge by facilitating the discovery, access, integration, interpretation, and analysis of distributed ecological information. In the hydrologic area, however, an equivalent data system is not available, although some software tools such as geographic information system (GIS) tools (e.g., Arc/Info and ArcView) are widely used to delineate watersheds, obtain river networks, and display spatial images. JOSS (Joint Office for Science Support) operated by UCAR also provides some preliminary tools for obtaining data different data sources. The USGS developed Modular Modeling System (MMS) utilities -- Object User Interface (OUI) providing a general framework to couple disparate environmental models and to manage some temporal and spatial data at limited scales (http://www.brr.cr.usgs.gov/projects/SW_precip_runoff/mms/). With recent support from the National Science Foundation, an effort to develop a system for the hydrologic community is now started through the Hydrological Information System of CUAHSI (Consortium of Universities for the Advancement of Hydrologic Science Inc.).

1.3. Problem Definition

The main focus of this research is to develop a data integration model and tool to facilitate easier access of various data sources which are autonomous in nature with heterogeneous data formats needed for improving widely hydrological research and application activities in hydrology community, and in particular, for hydrological model calibration processes at the NWS (National Weather service). The new model and tool will have the following four features:

(1). Various heterogeneous data sources with dramatically different formats and structures will be coherently managed, shared, and visualized with extensibility, scalability, uniformity, and transparency.

(2). The model will allow geographically distributed researchers to effectively and rapidly search, access and understand massive amounts of data by specifying query conditions, browsing, analyzing, aggregating, and visualizing the queried data, and customizing the formats of retrieved data.

(3). The model will allow users to analyze and visualize the accessed data first before retrieving them which could facilitate the data selecting process.

(4). The tool conforming to the model will be platform independent and have one consistent and unified user-friend interface, based on any type of web browser, such as Netscape or Internet Explorer.

1.4. Approach

We address the problems of integration with heterogeneous data sources and datasets using a novel tree based integration model supported on XML-based extended wrapper/mediator architecture. Our system is called as HIDE - Hydrological Integrated Data Environment. The model specifically addresses the data integration needs of autonomous systems (Sheth, 1998) considering the structural and volatile nature of the data. HIDE, assuming that datasets are external and physically distributed across a network and separate from our system, enables a loosely coupled architecture with a dynamic data fusion methodology. A schematic representation of the HIDE system is shown in Figure 1.1.

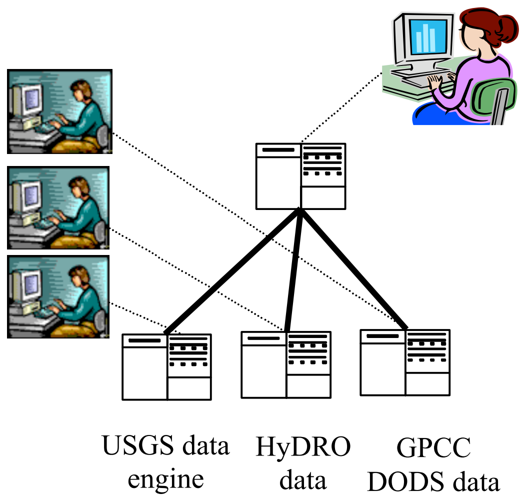


Figure 1.1 A Schematic representation of the data integration of HIDE

The aim of the model is to enable user to pose a query specifying what she/he wants to know and the system would locate where the relevant data is and present the integrated data to the user. For example, the user would be able to obtain data much more easily to answer questions such as "*What is the spatial distribution of annual precipitation for the year of 2003 around the world?*" or "*How does the spatial precipitation distribution in the past ten years compare to the climatology mean spatial distribution?*" or "*What are the precipitation trends (e.g., drying or wetting) in the Eurasia sub-continent?*" For instance, to answer the questions above, the researcher using HIDE would be able to conduct the following steps effectively and user-friendly:

(1) Find data source(s) by providing source names, attributes and domain concepts in the search engine;

(2) Extract useful data, for instance, "year 2003" by posting a query;

(3) Retrieve the data; store it in data storage mechanism;

(4) Repeat steps 1 to 3 for all other relevant data sources;

(5) Aggregate and fuse datasets from different data sources as needed;

(6) Reformat datasets to meet the data format requirement of the models and/or analytical tools she/he plans to use.

Clearly, performing each of these steps would require considerable amount of time and effort with current available / existing systems or software.

Most of the information systems deal with data from non-autonomous or semi-autonomous data sources. The autonomy nature of the data sources with respect to the management of data, data representation, semantic nature, implementation and sharing with the other systems plays a significant role in the architecture of the information systems. To fully address the autonomy nature of data sources, our model assumes each data source as a “black box”, while concentrating mostly on the data extraction technologies and thus handles the problem of autonomy in data sources (i.e., autonomous data sources).

This thesis contributes to the realization of a data integration model with dynamic data fusion capability. The model addresses the semantic and structural interoperability issues in autonomous data sources with unstructured or semi-structured metadata. Based on the discussions, the work proposes a suitable architecture and metadata standard for heterogeneous datasets and data sources from hydrology community with scalability, flexibility and uniformity which often other general hydrology information system suffer from. Finally the work involved the development of a web based system with features search engine, query engine, data retrieval and visualization methods.

1.5. Outline

The remaining of the thesis is organized as follows. Chapter 2 delves into the various issues of data integration of information systems and how one type of information system is different from another. Chapter 3 details our approach including the data integration model, metadata mechanisms, design objectives, and integration levels. Chapter 4 explains how a new data source or dataset can be registered to HIDE. Chapter 5 discusses the integration of a new data source from NWS (National Weather source) to HIDE. Chapter 6 summarizes our work and concludes with a synopsis of the future enhancements to the model.

Chapter 2 . Literature Review

Research on information systems and data integration issues can be generally classified into three categories: Issues of heterogeneity and its impact, architecture, and metadata standards. A brief introduction of the existing research in these areas is described in the following sections.

2.1. Interoperability and its Impact on Information Systems

Data and information interoperability has gained attention due to several reasons such as the easy access to the independently managed information sources due to excellent web interconnection and distributed computing architecture, reusability and sharing of information knowledge. In Geographic information systems, interoperability has been a major concern, as they deal with vast amounts of data used on a daily basis for solution on problems such as weather forecasting, development of forecasting models etc. Some of the key interoperability issues can be explained on autonomy and heterogeneity between the information systems.

Most of the information systems managed by entities such as USGS (US Geological Survey), NWS (National Weather Service) etc are autonomous in nature. Based on the reported work (Sheth, 1998); a data source participating in a federation can exhibit several kinds of autonomy such as design, communication, association and execution. The decision of the component to choose its own design in data representation, data management and data conceptualization falls into the category of design autonomy. Communication autonomy decides whether the component would like to share its data with others. In execution autonomy the component can make decisions on the changes that can happen locally without affecting other source's participation in the federation. Association autonomy implies that component can decide what amount of data can be shared with other components of the federation.

The complexities in data integration can also be attributed to various heterogeneities of the data as well as the data systems. One can define several kinds of heterogeneity leading us to several levels of interoperability (Sheth, 1998): system, syntax, structure and semantic. In this classification, the machine readable aspects of data representation falls into syntactic heterogeneity, and the representational heterogeneity in terms of data modeling falls into structural heterogeneity. Semantic heterogeneity relates to the difference in the semantics of the datasets to be relevant to semantic interoperability.

The semantic interoperability requires that system understands the semantics of the information request as well as the those of the information sources and satisfy the request as well as it can. The goal of a well defined data integration system is to construct a global schema of the data from multiple heterogeneous (syntax, structure, semantics and system) data sources. The global schema is a combined view of the underlying data sources and provides a uniform query interface to pose queries independent of the location, and heterogeneity of the data sources.

In a scientific data integration scenario, the heterogeneities between sources have a huge impact due to their autonomous nature. Boucelma et al., (2002), lists some of the key issues to be considered: the diverse nature of data sets in terms of structure, schema, formats and coverage, the methods of data publication and management and the volatile nature of the data sources. With the advent of 'Semantic Web' technology, there have been various efforts in the field of scientific data integration to improve the semantic interoperability with various data sources. Recent developments focus on the design of ontologies to support the semantic data integration. The advent of semantic web will lead to better data retrieval methods by incorporating data semantics into the search process.

Ontology can be described as a formal specification of a certain domain, a shared understanding of a domain of interest and a machine manipulable model of a domain of interest. Ontology can also be defined as "An explicit specification of a conceptualization" (Gruber, 1993). There are multiple logic based ontology languages such as OWL (Web Ontology Language), RDFS (resource description framework) for specifying ontology of the resources.

The reported work (e.g., Egenhofer, 2002, Lin et al., 2003, Fonseca et al., 1999, Uitermark et al., 1999, Gupta et al., 2002, Bowers et al., 2003) illustrates various efforts in scientific domain addressing the needs of ontology to resolve semantic data heterogeneity. The creation of a “Semantic Geospatial Web” (Egenhofer, 2002) requires the development of multiple spatial and domain ontologies and processing of geospatial queries against ontologies and retrieve results based on the match between the semantics of information need and the semantics of the information resources. This will enable users to retrieve more precisely the data they need, based on the semantics of the data. This can be exemplified based on an example. Suppose a user would like to conduct hydrological studies about the lakes in Maine. In order to retrieve the datasets, a query such as “*Find the precipitation datasets for lakes in Maine”* would be used by the user in an appropriate search engine. However if the datasets are distributed across multiple organizations and research institutions, with multiple characteristics such as “names of the lake”, “spatial distribution of hydrology data across United states” or the datasets uses counties to reference lakes, the search using the existing search engines would not be possible as they do not address the semantic needs of the request. Hence, without the semantic knowledge of datasets in scientific domain, it becomes a tedious approach for finding and analyzing the data. Moreover finding the appropriate datasets takes most of the time for the data analysts. Hence a data integration model should address the semantic requirements as well as the syntactic and structural nature of the data.

However, defining a comprehensive ontology for unstructured data in an autonomous data sources is a cumbersome process. Our model addresses this issue by defining a taxonomical domain model along with a structural model of the logical organization of the datasets and thus handling the semantic, syntactic and structural heterogeneity. The model also considers different aspects of the autonomy in design, communication, association and execution of the data sources.

2.2. Mediation

Efforts in the scientific data interoperability generally fall into two major categories; data warehousing and mediator-based systems. The data integration systems based on digital library initiative belongs to data warehousing categories and webservice based systems belongs to mediation category. Mediator based systems are constructed from a large number of relatively autonomous data sources, communicating using a standard protocol and enabling an "on-demand" data integration. Thus, the mediator approach provides scalability and modularity. A mediator is a software module that exploits encoded knowledge about some sets or subsets of data to create information for higher layer of applications. Hence it can hide all the heterogeneities over the underlying systems. Apart from these advantages, mediator based systems while responding to user queries, only retrieves and combines the query results from participating data sources without actually retrieving and combing the raw data.

The mediator based systems follows 3-level architecture (Wiederhold, 1992): a foundation layer with "wrappers," a mediation layer which supports varying querying capabilities of the wrapped data sources and an application layer. Given a user query against the global integrated schema, the mediator transparently decomposes it into local sub-queries to appropriate data sources, collects the query results, transforms into a global data model and presents to the user. The wrapper translates the request from the mediator's language to that of the information sources and transforms the results from the information sources back to the mediator's language. Acting as proxy of information, wrapper "talks" to the sources in its own language (native language/API) or interface and communicates with the mediator in a commonly agreed language. The wrapping of the data sources makes the diversity in protocols manageable. Also, the use of a global data model at the mediator layer enables modeling of systems with semi-structured or un-structured data. A mediator – wrapper system architecture is shown Figure 2.1.

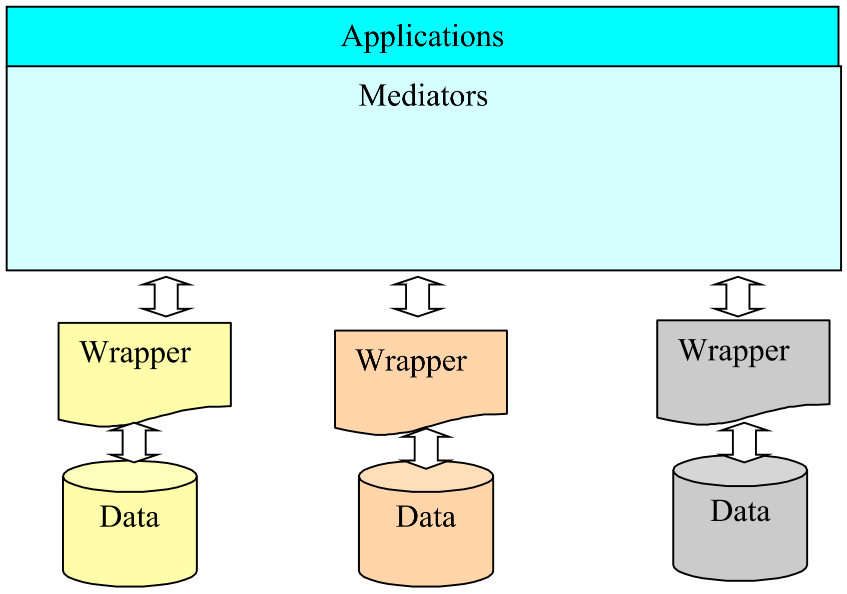


Figure 2.1 Wrapper – Mediator Architecture

Most of the mediation based systems are built on the structural knowledge of the datasets. However, with the introduction of semantic requirements, a semantics based mediation system is most appropriate to solve the integration needs of the scientific datasets. In a semantic mediation, sources cannot be integrated solely based on their structural schema but rather based on a conceptual model. There have been many efforts to use semantic mediation in the scientific integration scenario (Gupta et al., 2002). In their approach, a source registers with integrating system wrapped in its "conceptual model." The conceptual model of the source consists of its object model, local ontology, and contextualization of the local ontology relative to the mediator ontology. This approach requires a thorough and complete understanding of the semantics of the underlying datasets which may not be a possibility with autonomous data sources with unstructured datasets. Following along similar lines, we define a taxonomical based concept model of the source, with a wrapped logical schema and perform integration as a "tree." In order to facilitate the search mechanisms, we employ a keyword based search on the semantics of the datasets, by traversing down the tree to find the appropriate datasets. Mediation in our approach follows the semantic needs of the system along with the "wrapped" logical (syntax and structural) schema integration.

2.3. Example Information Systems

Information Systems can be broadly classified (Sheth, 1992) based on interoperability issues to First Generation, Second Generation and Third Generation information systems. Much of the work in First generation systems occurred in multidatabases or federated databases. The emphasis in this generation was to achieve system interoperability particularly addressing the heterogeneity arising due to differences in DBMSs. The second generation of information systems supports a wide variety of data including visual media, uses metadata to support interoperability and integration along with knowledge representation and reasoning. The third generation information systems deals with more autonomy, more heterogeneity and more digital data, but also operations and computations (simulations) that can create new data. The key challenges in this generation are the understanding of the system at the semantic level where the systems help people at the information level and not at the data level. Table 1 provides an overview of generation of information systems based on a variety of criteria.

	Generation I	Generation II	Generation III
Types of interoperability emphasized	system (computer system and communication); limited aspects of syntax and structure (data model); transparency of location, distribution, replication, data models	syntax (data types and formats), structure (schematic, query languages and interfaces)	Semantic (increasingly domain-specific)
Dominant interoperability architecture	multidatabases or federated databases	federated information systems, mediator	Mediator, information brokering
Scope of system interoperability	handful of interconnected computers and databases	tens of systems on a LAN, databases and text repositories	enterprise-wide and global scope
Communication infrastructure on which system interoperability solutions are built	proprietary (IBM domination), TCP/IP	TCP/IP, http, CORBA	Internet/Web/Java, distributed object management, component, but increasingly higher-level such as multi-agent, mobile
Types of data	structured databases and files	structured databases, text repositories, semi-structured and structured and data in generic (e.g., SGML, HTML) and domain specific formats	all forms of digital media with increasing support for visual/spatio-temporal/ scientific/engineering data;

Data/ information interoperability approaches	structural and data model, data representation	understanding of a variety of metadata, comprehensive understanding of schematic heterogeneity	comprehensive use of metadata, increasing emphasis on semantics and ontology supported approaches
Access options	database query language (SQL) for structured databases, keyword accesses for textual data/files	keyword-based attribute and (limited) content-based access, (limited) ontology-based access,	multimedia views; visual interfaces; information requests that are media-independent, multi-ontology based, context-sensitive and domain-specific
One representative complex query	Find a four star restaurant with less than $25 average cost that serves Mediterranean food in Richmond (a multidatabase query on distributed structured databases)	Find flowers suitable for winter gardens that look like *this* <image> with a soft smell (a keyword-, attribute-, and content-based query on text and image data repositories)	Find a block of land with urban land cover and moderate relief and population greater than 5000 and area greater than 1000 sq ft suitable for a strip mall (a query with terms whose meanings are understood by the system, and may involve multi-step processing against multi-modal data)

Table 1 An Overview of three generation of information systems

A representative set of information systems that support access to heterogeneous sources includes TSIMMIS (Hammer et al., 1995 & 1997), Information Manifold (Levy et al., 1995 & 1996), InfoHarness (Shklar et al., 1995), SIMS (Arens et al., 1993&1996), Infosleuth (Bayardo et al., 1997), OBSERVER (Mena et al., 2000), MIX (Zaslavsky et al., 2000, Baru et al., 1999) and DISCO (Tomasic et al., 1998). SIMS, InfoSleuth, OBSERVER and MIX belong to third Generation of information systems as they support ontology with knowledge based reasoning.

The focus of research in TSIMMIS is to combine data from structured and unstructured data sources which are encapsulated using wrappers and an automated generation of wrappers for the data sources based on templates. The wrappers translates the data model to a global data model while the mediator layer routes the query requests to appropriate data sources. The query is formulated using a rule based query language called MSL. The data is represented using Object Exchange Model (OEM). The MSL query extracts the OEM objects and its sub objects by matching patterns in the query against existing OEM structures (wrapper templates). TSIMMIS deals with data in a relational database system.

In the Information Manifold, the information is integrated from disparate (structured and unstructured) data sources. The architecture is based on a knowledge base built by describing the source capabilities and uses views to express the limited capabilities of the sources. The system presents the user with a unified view of the information space called *world view.* The world view is expressed in an object - relational data model. The queries can be posted as “declarative” in terms of objects and relations in the view. Every information site expresses its contents using a site description language which would be used by the query processor to match the query. The site descriptions can be used to relate the semantic content of the information site to world view relations. The system also uses an optimized approach to connect to only required information systems in order to satisfy the query. This is achieved by using constraints in site relations and pruning site relations which are irrelevant to the query. The information manifold system can be used to integrate information from relational databases, object oriented knowledge base etc.

The InfoHarness system provides integration of web accessible documents and relational databases with the support of third party indexing and browsing technologies. The SIMS system uses an application domain model, describing the properties, relationships between objects. A model is constructed for each information source, involving the source and its relation to domain model. The user formulates queries using terms from application domain without actually knowing anything specific about the information sources.

The Infosleuth system uses a common domain ontology which gives a declarative description of the semantic information without the underlying syntactic representation. The local database schemas are being mapped to the domain ontology. The systems utilizes an agent based approach, for representing each information resources and semantically match information needs with currently available resources using brokers and so, retrieval and update requests can be routed only to relevant resources. The system consists of a network of cooperating agents such as User Agent, Ontology agent and Broker agent. The agents advertises their services and process the requests based on the Infosleuth ontology or route to the appropriate agents or decompose the queries into a collection of sub-queries and forwards them to appropriate agents. The “Infosleuth ontology” describes the relationship between agents and their knowledge. The decomposition of query is supported on the domain ontology chose by the user. The agents communicate between each other using a specialized communication language. The different types of agents in the architecture are User agent, broker agent, ontology agent, resource agent, data analysis agent and task execution agent.

The OBSERVER project presents a query processing architecture for the Global Information Systems. The user of the OBSERVER can browse various domain ontologies which are the conceptual view of the Information content of heterogeneous repositories. The interoperation between ontologies is achieved through synonym relationships between various terms across ontologies. The key objective of the approach is to reduce the knowledge of structure and semantics of data in huge number of repositories to small, by knowing the synonym relationships between terms across ontologies. The query presented by the user is translated to a query, represented as a conjunction of constraints expressed using description logic based on the ontology chosen by the user. Then the query processor translates the query in terms of the ontology of the information sources and thus preserving the semantics of the user query. If the user is not satisfied with the results, he/she can chose other ontologies and proceed with the query. The systems make the assumption that an elaborate ontology for the components is available. This approach is mostly used when the data is represented as a "document" where generation of ontologies is not entirely limited by the domain.

The DISCO project addresses the issues of "fragile mediator" problem due to the volatility of the data sources. If there is a tight integration scheme, then whenever a new data source is added, there is an added responsibility on the DBA to make sure that the integration goes smoothly with other data sources. This is handled by providing a partial query evaluation scheme that accounts for source unavailability and thus managing graceful failures of the sources.

The MIX project is an XML based mediation architecture for spatial data interoperability in Geographic information systems. The MIX architecture uses XML wrappers for individual data sources to export data in a uniform format to the mediator. MIX uses a lazy approach or on-demand approach i.e. XML queries are rewritten at runtime as they flow downwards from user to sources. An interesting aspect of MIX is that it uses a spatial mediation engine to handle spatial interoperability and thus enabling spatial data integration. The communication between various modules in MIX is based on XML. The spatial mediator parses the spatial part of the query and generates an evaluation plan. MIX can dynamically evaluate the minimal number of sources required to satisfy the query attributes by exporting the schema information by the wrappers as XML DTD. This helps if the schema is not known at the time of integration.

Coming to the Hydrology domain, the system HDMRAS (Hydrologic Data Management Retrieval & Analysis System) (Liu et al., 2003) is another example of integration and management of heterogeneous data for hydrologic studies. HDMRAS uses a metadata based approach for data integration. The system, however, assumes that all datasets being integrated are not physically distributed over the network, but reside on local data storage. A user in HDMRAS can define various characteristics such as data organization, structure, and semantics of the datasets to be integrated through the metadata files. HDMRAS dynamically generate the query interfaces written as metadata files, create and execute a query evaluation plan and retrieve the results. HDMRAS demonstrates the effective usage of metadata as a primary task in scientific data integration scenario. But, as a part of the data source registration process, datasets are to be uploaded into the system, which imposes an extra responsibility of data management into the system.

2.4. Metadata Standards

The metadata standards play a crucial role in a successful integration of data. Metadata is usually defined as “data about data.” But often it is more than that, involving information about the data representation, management and semantics. The metadata may provide the information related to content, context, quality, structure, accessibility etc. of a specific dataset. It necessarily offers comprehensive information about the data without completely describing it. Metadata descriptions present two advantages: an abstraction of the representational details of the data and capture the information content independent of the representation of the domain knowledge describing the domain to which the data belongs. Some classifications of metadata (Sheth, 1992) are, Content independent metadata (e.g.: type of sensor used to record an image), content dependent metadata (e.g.: size of the dataset), direct content-based metadata (e.g.: indices based on the text of the document), content-descriptive metadata (e.g.: textual annotations describing an image), domain independent metadata (e.g.: HTML document independent of the subject) and domain specific metadata (e.g.: land cover from a GIS domain). Domain specific metadata are particularly useful as they utilize the semantic understandings of the dataset in an effective data retrieval process. Information systems use metadata for various responsibilities.

Foundational to our approach is the use of metadata for describing data sources, datasets, query plans and transformation of source data model to our data model. The diversity of efforts in the earth science community with respect to metadata standards is many. The Federal Geographic Data Committee Standard (FGDC) has designed geospatial standard CSDGM which is a content standard for Digital Geospatial Metadata. The Earth Science Markup Language (ESML) describes the structure, semantics and content of any Earth science dataset in any data format. The ESML provides syntactic metadata describing the data in bits and bytes. The data sets are modeled as binary/ASCII sequence, arrays and structures.

The Geography Markup Language of OGC (Open Geospatial Consortium) describes the features and geometry associated with data sets. It is a widely supported specification for geographic information. The specification is based upon a large number of other W3C, IETF, ISO, and OpenGIS standards. GML uses a wide variety of objects to describe the geography including feature, coordinate reference systems, geometry, topology and time. GML specification describes an open framework for defining geospatial application schema and thus increasing the ability to share the geographic application schemas and the information. The GML emphasize on the geographic nature of the data and, needs to be tailored for other domain needs.

In the Ecology domain, Ecological Metadata Language (EML) describes the ecology datasets. EML offers modularity as it is designed as a collection of modules. The intent is to provide a set of core modules which can be later customized without any lengthy approval process. Although EML presents a detailed structure of the data, it has strived to strike a balance between too much of information and too little information. EML offers compatibility as it has been developed from other geographic standards.

In all the above mentioned XML standards, it has to be noted that, standards describe the data and datasets with little information about data source. EML tries to solve this to some extent.

Considering the importance of the information about the sources in a data integration scenario, in the Sensor network domain, the standard SensorML (Sensor Modeling Language) offers the model and schema for defining the geometric, dynamic and observational characteristics of sensor. These sensors that are capable of observing and measuring different physical and chemical properties, has specific response characteristics that can be used to assess the values of the measurements and quality of these measurements. The purpose is to give sensor information in support of data discovery and quality characteristics of the data. An interesting aspect to note that SensorML provides the performance characteristics (threshold, accuracy) of the measurements, and archives the fundamental properties and assumptions regarding the sensor. SensorML gives a functional model of sensor, not a detailed descriptions of the hardware.

From all the above mentioned standards, it is quite clear that metadata plays a significant role in the data integration as a solution to heterogeneities across sources and architecture.

Chapter 3 . Data Integration Model

3.1. Introduction

Heterogeneity can be due to many technological differences between information systems. One aspect of heterogeneity is based on syntax, structure and semantics. Differences in the formatting of the data lead us to syntactic heterogeneity. Structural heterogeneity revolves around the data modeling constructs. Differences in the representation of the domain knowledge or concepts relates to the semantic heterogeneity. An Absolute data integration model should handle all the issues of various heterogeneities.

HIDE assumes that the data sources and datasets are distributed physically across networks with various methods of accessibility. The data integration model of HIDE logically integrate these data sources and datasets, thus presenting the user with a unified view of the underlying data irrespective of the geographic location, data formats ,low level data models and semantics of the data. To address the issues of heterogeneity, HIDE employs a tree based data integration model incorporating the semantic knowledge as well as the structural knowledge of the data sources and datasets. The syntactical information relevant to the data sources are captured in the metadata model of the HIDE. The hierarchical nature of the data integration model facilitates scalability, extensibility, modularity, user management, security and privacy.

3.2. DataNode(s) and DataNode Tree

The model is constructed by disseminating the data and information, into information units, called DataNodes while keeping in mind the design objective of simplicity. We associate each unit with information structure, semantics, syntax, and contents which facilitates data analysts to perform any useful query.

A DataNode can represent a data source such as a sensor node, a data file, or data organization (US Geological Survey), a data portal/engine and a hydrology concept. The DataNode is the smallest entity of the system and a collection DataNodes conceptualizes a physical organization of data, a logical organization of data or a user specific view of the data organization. This level of abstraction provides the user with flexibility and defines a clear line of separation between the low level and high level view of the data.

Once the DataNodes of various views has been identified, users can define the relationships among them. We describe the relationships between DataNodes as links in a tree, finally evolving into a DataNode tree. The relevance of the relationships between each DataNode is reflected differently in each view of the tree. This hierarchical dissemination of the data access space into a tree helps a better classification and understanding of the data. A typical example of a DataNode tree is shown in Figure 3.1

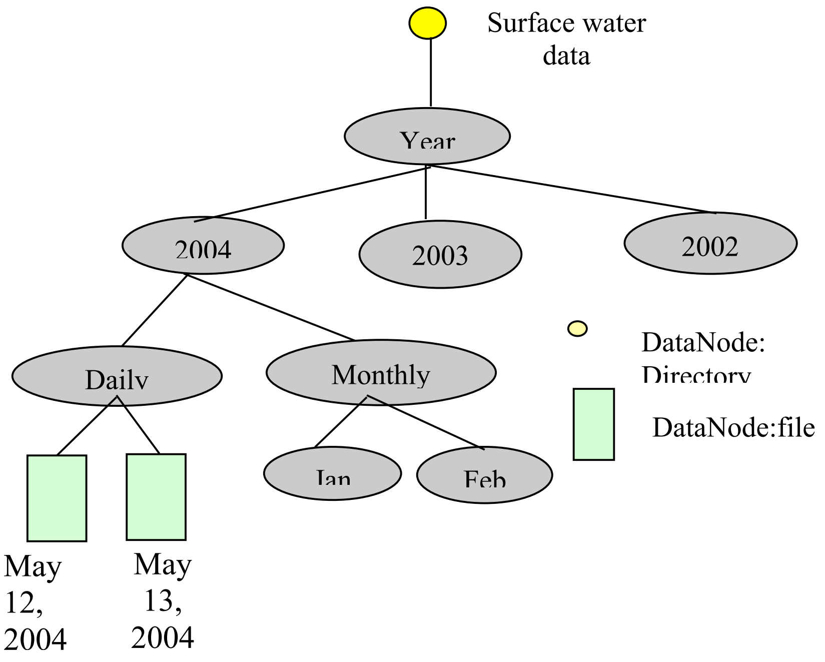


Figure 3.1 A DataNode Tree representation

3.3. Views – User, Logical & Physical

The data integration model is also separated into three views. The semantic nature of the hydrological datasets is modeled as a domain model and acts as the user view of our integration model. The user view provides a taxonomical nature of the concepts in the hydrology domain. The structural information of various data (structured, semi-structured and unstructured) which is hidden in most of the autonomous systems is presented as structural model in the logical view. The logical view can be looked upon as a logical organization of the data resembling a tree as perceived by an integrator. The combination of the domain view and logical views of the data sources and sets forms a DataNode tree. The physical organization of the data is defined by the physical view which is mostly hidden in the data sources.

The relationship between various DataNodes in the DataNode tree is identified based on the view to which it belongs. In the user view (domain level), the relationship is more of a taxonomical nature. While in the logical view (structural view), the relationship is more of a structural nature.

Three kinds of users are recognized in the context of data integration: Users as domain analysts who query the data and perform data retrieval, analysis, fusion and mining; Users as system integrators who integrate an external data engine into the HIDE system; And the users assumed to be part of the external data engine who perform the low level data management such as data file organization. Only the first two categories of users are associated with the HIDE system. The conceptual view of the DataNode tree for each kind of users is different. This difference in views of the system is reflected on the views of the model: the physical view as seen by the low-level data managers, the logical view as seen by the HIDE system integrators, the user view as seen by the high level domain analysts. As the physical view is associated with the external data engine unit, it is assumed to be separate from the HIDE and is entirely limited within the data engine unit. A typical example for the three views is shown in Figure 3.2.

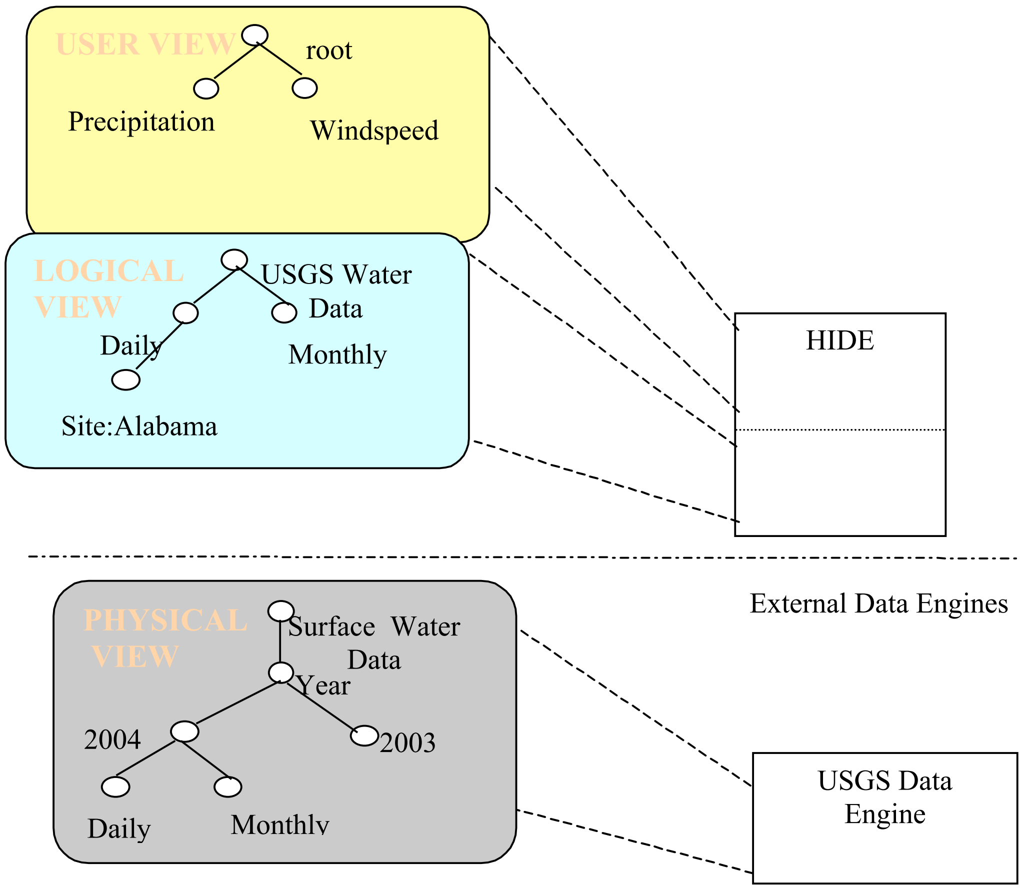


Figure 3.2 The three views- user view, logical view, and physical view and the corresponding DataNode tree.

Chapter 4 . Metadata model

4.1. Introduction

Metadata standards are crucial part of an Information system. Metadata which is typically ”data about data” provides comprehensive information about the data. Or in other words, understanding of the metadata of a data/resource is sufficient to understand the data/resource. For example; Metadata for a library contains the information of the various titles, authors, abstracts of book or articles. While the metadata for a geographical map contains information about the scale of the map, geographical coverage etc. Note that, based on the nature of the data and the context, the metadata for data/resource/datasource can vary. Hence metadata for data ‘must’ contain all the relevant information about the data, and not the data itself.

The introduction of XML has revolutionized the scope of the representation of the metadata. The extensible Markup Language (XML) offers an ideal means for an effective representation of the metadata and information exchange across domains. In simple terms, an XML document is a text based metadata for a data/resource/datasource. “Schema” has been used to provide a means for defining the structure, contents and semantics of the metadata. Hence schema is a model of a metadata or a template for a metadata, defining the rules for the metadata. Therefore to understand all the aspects of the data, one requires metadata which follows the rules of the schema. A simple representation of the relationships between data, metadata and schema is shown in the Figure 4.1

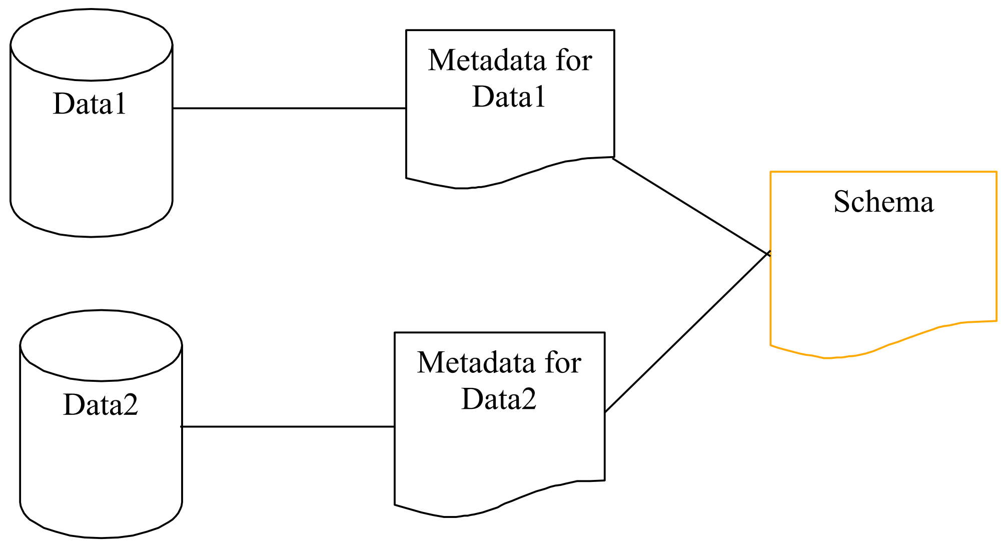


Figure 4.1 A simple representation of the relationship between data, metadata and schema. Every data has a metadata associated with, which follows the rules of the schema.

In this thesis, a metadata schema is referred as a metadata model. Considering the relevance of metadata in a data integration system, the metadata model of HIDE encompasses various requirements of integration. The necessary information required for integrating a data source/ data set into HIDE system as a DataNode is extracted through a metadata model. We define that an effective integration, requires the information to answer the following questions.

1. Describe the data source or dataset.
2. What kind of operations is permitted on the data source / dataset?
3. What are the syntax and semantic details of the data from the data source?
4. How to access the external data engines?
5. What kind of transformations should be applied to the data to transform to the data model of HIDE?

In the HIDE system, appropriate information as answers to the above questions for each data node which effectively represents a data source/dataset, is provided as metadata. Using metadata to define the data source and its operations and accessibility enables handling the interoperability efficiently between data engines. Our primary aim in designing the metadata was to define a certain level of flexibility and simplicity to the users in integrating the data sources/datasets. Once the user as an integrator presents the metadata to the HIDE, the data source specified by the metadata can be integrated without any modifications/changes to the HIDE system. We use XML language for the metadata representation.

HIDE defines a clear separation between the description of the data engines and the low level details of accessing the data from the data engines. We separate the information with respect to the DataNode as description model and the operational details of the node in an operational model. The operational model covers the details of the operations, transformation, accessibility and syntactical information. Hence there exists a dependency between the operational model metadata and access, transformation and syntax metadata. Also the existence of the operational model depends on the existence of description model. The organization of the metadata for DataNodes is shown in Figure 4.2

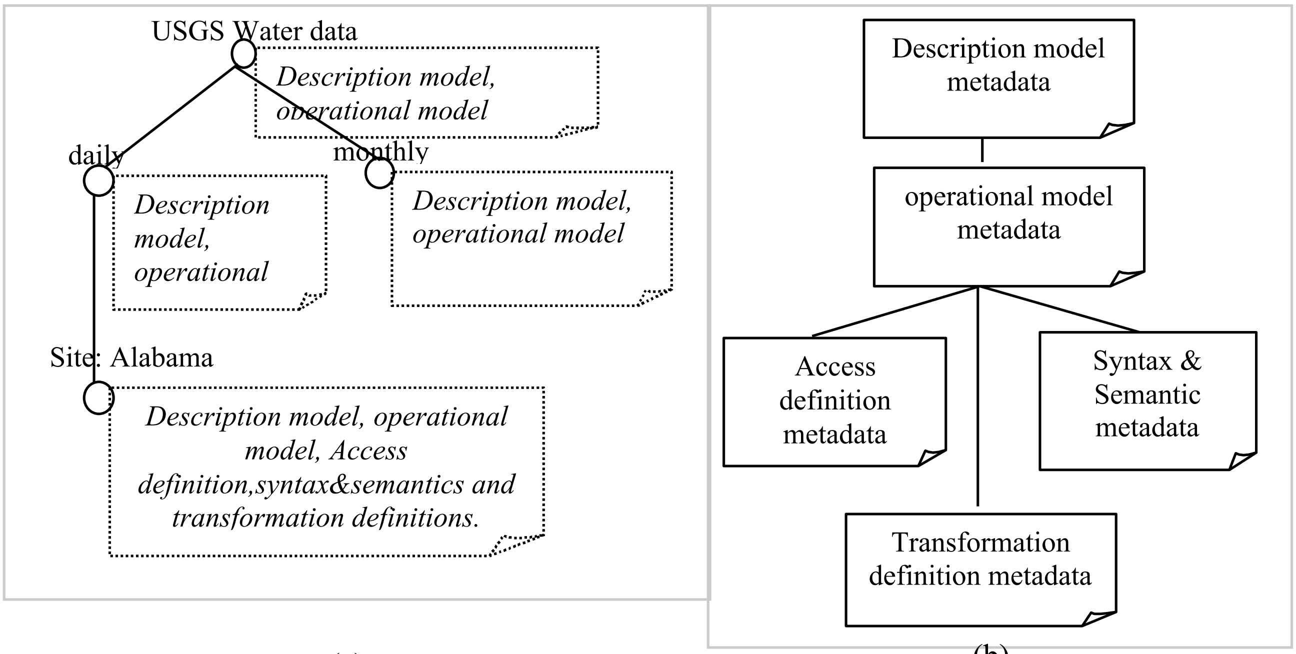

USGS Water data
Description model, operational model
daily
Description model, operational
monthly
Description model, operational model
Site: Alabama
Description model, operational model, Access definition,syntax&semantics and transformation definitions.
(a)
Description model metadata
operational model metadata
Access definition metadata
Syntax & Semantic metadata
Transformation definition metadata
(b)

Figure 4.2 The metadata representation for a DataNode tree. A) The metadata required for each DataNode. B) The dependency between various metadata.

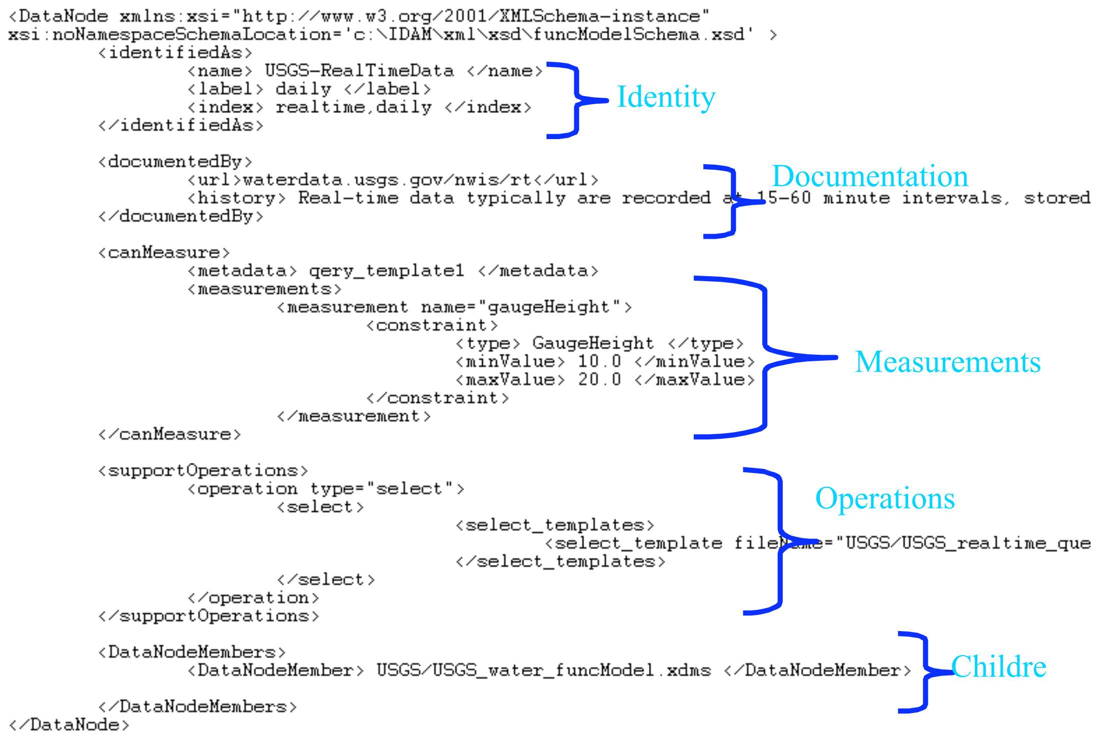
As shown in Figure 4.2, the description model metadata is to be specified for all the data nodes as it defines the DataNode. The operational model is optional for metadata except for leaf data nodes. All the other metadata files are dependent on the existence of the operational model. Each metadata model is described in detail in the following sections.

4.2. Metadata schema describing DataNode:

Each DataNode in our system irrespective of view describes itself in an XML description model (XDMS). The details of the description model are captured in a metadata schema and are used by HIDE for defining the DataNode tree. This description model details the following information.

1. Identity: The identity for the data node. The system integrator can specify a unique id, list of indices for choosing the DataNode while performing a search and an associated name.
2. Documentation: The historical information about the data source / dataset that identifies the DataNode along with external links for additional information.
3. Measurements: The type of measurements and its quality constraints for a DataNode as a data source such as sensor node.
4. Children: A list of its child DataNodes for a specific data node.
5. Capabilities: The operational abilities of a data node as a data engine such as query operations, and data update operations.

A snapshot of the XDMS metadata file complying XDMS schema is showed in figure 4.3. See Appendix for the XDMS schema.



```
<DataNode xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation='c:\IDAM\xml\xsd\funcModelSchema.xsd' >
        <identifiedAs>
                <name> USGS-RealTimeData </name>
                <label> daily </label>
                <index> realtime,daily </index>
        </identifiedAs>

        <documentedBy>
                <url>waterdata.usgs.gov/nwis/rt</url>
                <history> Real-time data typically are recorded at 15-60 minute intervals, stored
        </documentedBy>

        <canMeasure>
                <metadata> qery_template1 </metadata>
                <measurements>
                        <measurement name="gaugeHeight">
                                <constraint>
                                        <type> GaugeHeight </type>
                                        <minValue> 10.0 </minValue>
                                        <maxValue> 20.0 </maxValue>
                                </constraint>
                        </measurement>
        </canMeasure>

        <supportOperations>
                <operation type="select">
                        <select>
                                        <select_templates>
                                                <select_template fileName="USGS/USGS_realtime_que
                                        </select_templates>
                        </select>
                </operation>
        </supportOperations>

        <DataNodeMembers>
                <DataNodeMember> USGS/USGS_water_funcModel.xdms </DataNodeMember>

        </DataNodeMembers>
</DataNode>
```

Figure 4.3 A snapshot of the XDMS metadata. A high level representation of the description metadata is shown.

4.3. Metadata schema defining the operational characteristics-query plan of a data node:

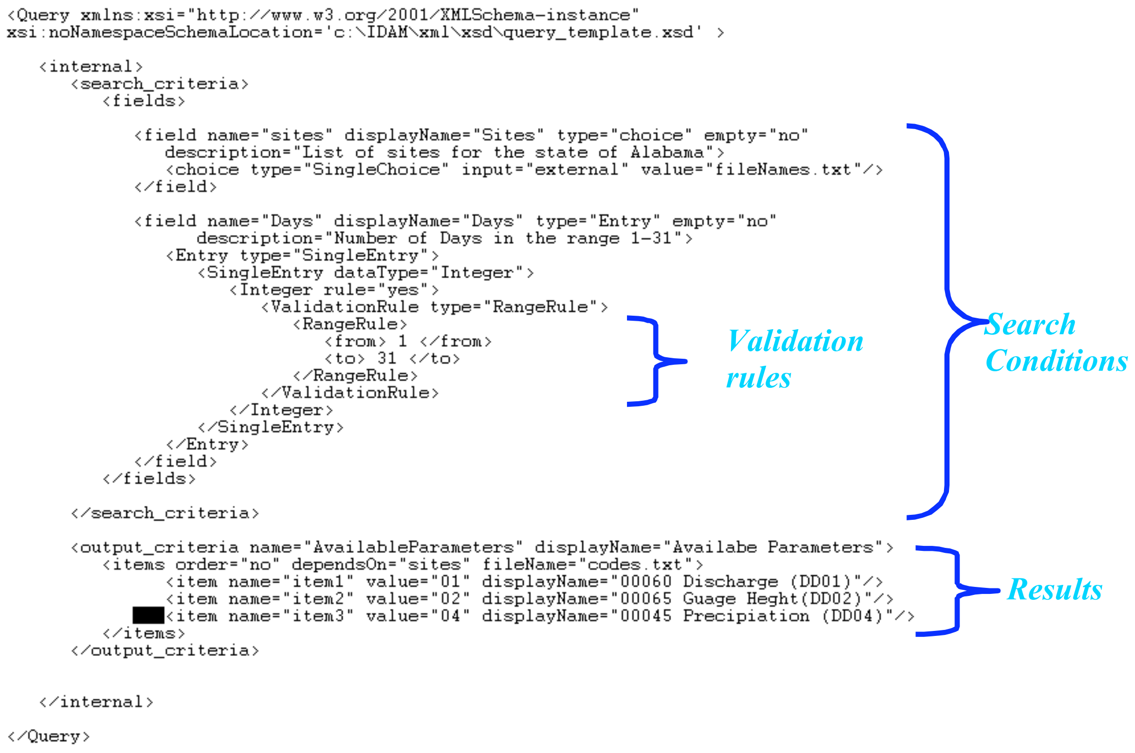
Disparity of interfaces to each data engine makes easy locating and access to data cumbersome. Hence our primary concern is to provide a uniform representation of the access interface to the user irrespective of the complexities of underlying data engine. This is achieved with the definition of operational model and is defined through the metadata schema XOMS (XML operational model specification).

A user as a data analyst typically performs two kinds of operation on the data, a data query such as "what is the wind speed for a period of 2 days in the state of Alabama" and a data update operation. Our system identifies that, these operations are performed on the information units – DataNodes. Each DataNode is provided with relevant operational information. The operations suggested by the analysts are translated to operations on the data node through the operational characteristics defined for each node. Our system specifically concentrates more on the query operations that can be performed on the DataNode irrespective of the view (user view or logical view) to which it belongs. The query operational characteristics of each DataNode is described as a query model which enables for an efficient tracing of the query from the root DataNode in the user view to the end leaf node in the logical view. The end leaf node of the query trace connects to the external data engine, possibly the physical view, performs the query and retrieves the result.

The operational model details the following information.

1. Conditions & Results: A data query operation can be considered as a combination of a set of search conditions and the result parameters. In the above example, search condition is "period of range of 2 days/ 2 months" and result parameter is "precipitation." In our system, user can define the possible search conditions and a set of result parameters in the query metadata for each node. Each search condition is identified as a field with a unique name, display name, display format, unit and a field type associated with it. The field type can be used to decide the kind of user interface applicable for a search condition.
2. Validation rules: Validation rules provide necessary information to the operation engine for performing validation on each field of the search condition. Examples of validation rules can be Range rule, format rule, date rule etc.

An instantiation of the model as a metadata, details the above information enabling an easy and flexible dynamic generation of the user interface. A snapshot of the XOMS metadata complying the rules of the operational model metadata schema specification is shown in Figure 4.4. See Appendix for the XOMS schema.



```
<Query xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation='c:\IDAM\xml\xsd\query_template.xsd' >

   <internal>
      <search_criteria>
         <fields>

            <field name="sites" displayName="Sites" type="choice" empty="no"
               description="List of sites for the state of Alabama">
               <choice type="SingleChoice" input="external" value="fileNames.txt"/>
            </field>

            <field name="Days" displayName="Days" type="Entry" empty="no"
                  description="Number of Days in the range 1-31">
               <Entry type="SingleEntry">
                  <SingleEntry dataType="Integer">
                     <Integer rule="yes">
                        <ValidationRule type="RangeRule">
                           <RangeRule>
                              <from> 1 </from>
                              <to> 31 </to>
                           </RangeRule>
                        </ValidationRule>
                     </Integer>
                  </SingleEntry>
               </Entry>
            </field>
         </fields>

      </search_criteria>

      <output_criteria name="AvailableParameters" displayName="Availabe Parameters">
         <items order="no" dependsOn="sites" fileName="codes.txt">
               <item name="item1" value="01" displayName="00060 Discharge (DD01)"/>
               <item name="item2" value="02" displayName="00065 Guage Heght(DD02)"/>
               <item name="item3" value="04" displayName="00045 Precipiation (DD04)"/>
         </items>
      </output_criteria>

   </internal>
</Query>
```

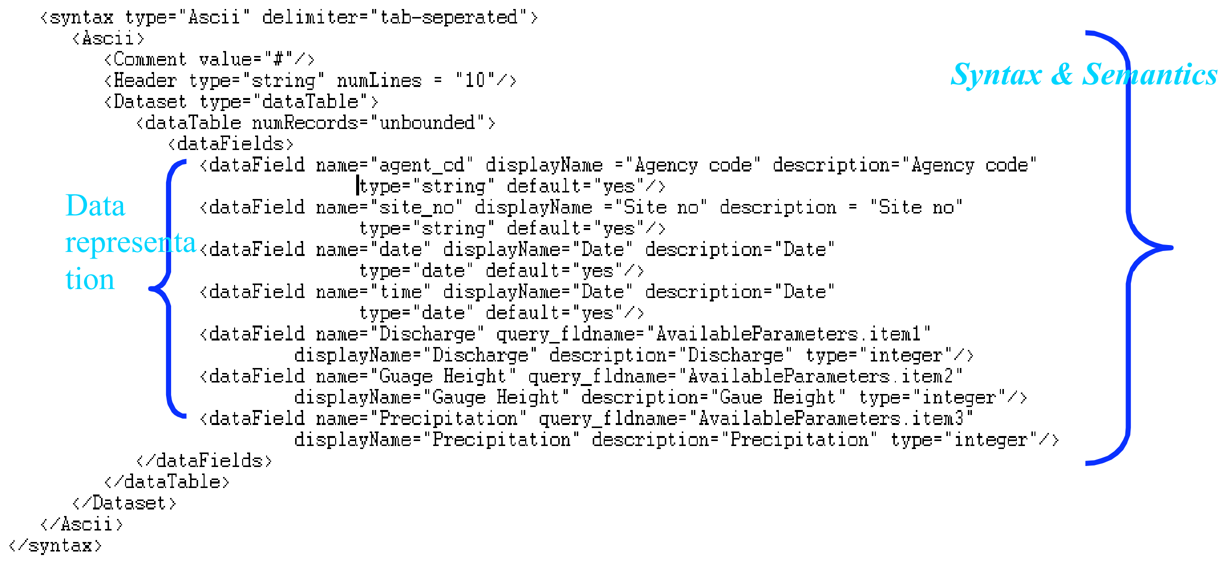
Figure 4.4 A snapshot of the XOMS metadata for precipitation data from USGS data source is shown. Each search criteria is associated with a field and a validation rule.

4.4. Metadata schema for describing the data (syntax):

Once the data has been retrieved by the HIDE system, a question that arises is "How the system can understand the data file." This is achieved by specifying the syntax information of the data retrieved by the system through XFD (XML format description) specification and can be represented through the XFD schema file. The schema describes syntax information such as (1) what is the format of the file ASCII/binary? (2) What is the type of the file such as comma-seperated/tab-sperated? (3) What are the retrieved fields? (4) What are the data types of the retrieved fields?

It is not necessary to define the metadata complying the XFD schema for all the retrieved data if they are from the same DataNode. In such a case, all such retrieved data can share single metadata file. It contains structural segments which further describes the details of the data structure. Binary metadata describes a binary format and ASCII metadata describes an ASCII format. Every data file begins with a header and comment (optional).

The data model representation of the data in the data file varies depending on the nature of the data. HIDE identifies data represented as a structure (binary) of data fields with varying types, table of data fields (ASCII) with varying types and an array of data fields. Each data field is characterized by name - A unique value which identifies the data field, display name- A name used for display purposes, data type - The type of the data field (integer, float, spatial array, and string), default nature of the data field and the name of the query field corresponding to the requested data field. A snapshot of the XFD metadata file which complies to the XFD schema is shown in Figure 4.5. See Appendix for the XFD schema specification.



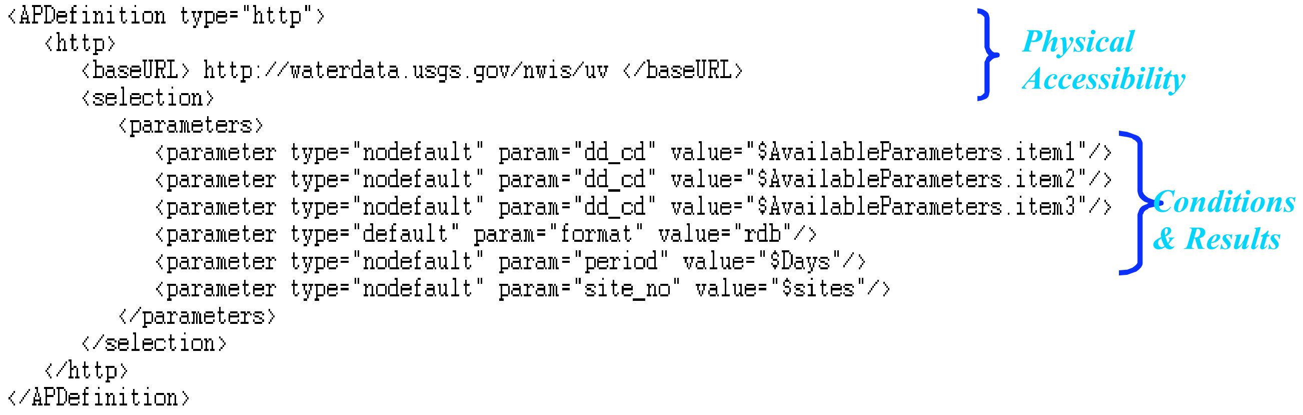
```
   <syntax type="Ascii" delimiter="tab-seperated">
      <Ascii>
         <Comment value="#"/>
         <Header type="string" numLines = "10"/>
         <Dataset type="dataTable">
            <dataTable numRecords="unbounded">
               <dataFields>
                  <dataField name="agent_cd" displayName ="Agency code" description="Agency code"
                               type="string" default="yes"/>
                  <dataField name="site_no" displayName ="Site no" description = "Site no"
                               type="string" default="yes"/>
                  <dataField name="date" displayName="Date" description="Date"
                               type="date" default="yes"/>
                  <dataField name="time" displayName="Date" description="Date"
                               type="date" default="yes"/>
                  <dataField name="Discharge" query_fldname="AvailableParameters.item1"
                         displayName="Discharge" description="Discharge" type="integer"/>
                  <dataField name="Guage Height" query_fldname="AvailableParameters.item2"
                         displayName="Gauge Height" description="Gaue Height" type="integer"/>
                  <dataField name="Precipitation" query_fldname="AvailableParameters.item3"
                         displayName="Precipitation" description="Precipitation" type="integer"/>
            </dataFields>
         </dataTable>
      </Dataset>
   </Ascii>
</syntax>
```

Figure 4.5 The figure shows XFD metadata for an ASCII comma-separated data file with a string header and the dataset. The dataset is of type dataTable with dataFields. Each dataField is further characterized with name, display Name, description and data Type.

4.5. Metadata schema for describing the access to the data engines:

One of the primary objectives of the HIDE system is to hide the low level details from the high level users (data analysts). However the HIDE system requires information on "how to access the external data engine." This information can be described through a XAD (XML Access point definition) specification and is represented by an XAD schema. The schema describes how to connect to the external data engine, submit the query and retrieve the results. A point to be noted here is that the XAD metadata complying the XAD schema is defined only for the end leaf node in the logical view as it makes the connection to the external engine and retrieves the results. An assumption made here is that all the external data engines provide an interface for accessing the data. The information provided by the specification is detailed below.

1. Physical accessibility: This information includes the protocol to be used for accessing the external engines, and the necessary additional information related to protocol. For example, if the protocol to be used is http, the additional information to be specified consists of the links to the external data engine.
2. Conditions & results: This information defines how the conditions and results specified by the user in the operational model can be transformed to the details required for the access to the external engine.
A snapshot of XAD metadata complying to XAD schema is shown in Figure 4.6. See Appendix for the XAD schema.



```
<APDefinition type="http">
   <http>
      <baseURL> http://waterdata.usgs.gov/nwis/uv </baseURL>
      <selection>
         <parameters>
            <parameter type="nodefault" param="dd_cd" value="$AvailableParameters.item1"/>
            <parameter type="nodefault" param="dd_cd" value="$AvailableParameters.item2"/>
            <parameter type="nodefault" param="dd_cd" value="$AvailableParameters.item3"/>
            <parameter type="default" param="format" value="rdb"/>
            <parameter type="nodefault" param="period" value="$Days"/>
            <parameter type="nodefault" param="site_no" value="$sites"/>
         </parameters>
      </selection>
   </http>
</APDefinition>
```

Physical Accessibility
Conditions & Results

Figure 4.6 A snapshot of the XAD metadata. The figure shows an Access point definition for a web based data source. The URL for connecting to the data source is represented as a baseURL along with the selection parameters. Each parameter has a name (param) and value associated with it.

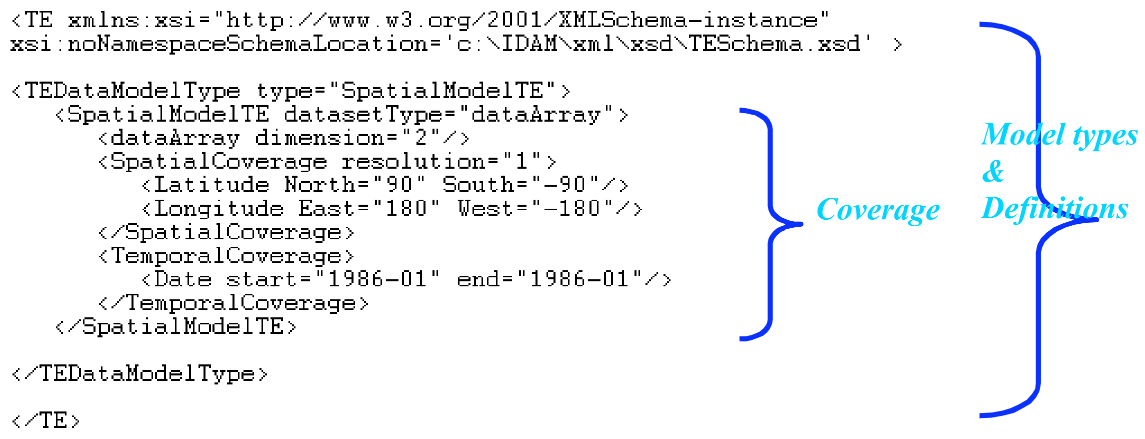
4.6. Metadata schema for transformation of the data:

One of the primary objectives of our system is to provide different data analysis and visualization techniques of the retrieved data to the external user. Due to heterogeneous nature of the retrieved data from various data sources/engines, an efficient method is required for transforming the data into a uniform data model. We have identified two basic kinds of low level data model for scientific data sets, a temporal model and spatial model. The necessary information for transforming the data such as the low level data model is defined in a transformation specification and is represented by XTMS (XML transformation specification) schema. The information is detailed as below.

1. Model types: Depending on the nature of the retrieved data from a data source, the integrator can specify what kind of data model is applicable for the data. As mentioned, user can specify two basic kinds: Temporal model, and Spatial model. User can also specify the combination of the two basic models: Temporal-Spatial model.

2. Coverage: Describes the coverage of data in terms of time, and space. This coverage (temporal and spatial) represents the extent of applicability of the data in those domains. The Temporal coverage defines a single time instance or a period of time for the dataset. The spatial coverage defines the area represented by the dataset. For example, the temporal coverage for a data node can be a period of time 1986-01 to 1986 -01, which indicates the month of January for the year 1986. The spatial coverage for a dataset can be 90° N to -90° S and 180° E to -180° W.

A snapshot of the XTMS metadata complying the XTMS schema is shown in Figure 4.7. Please see Appendix for the metadata schema.



```
<TE xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation='c:\IDAM\xml\xsd\TESchema.xsd' >

<TEDataModelType type="SpatialModelTE">
   <SpatialModelTE datasetType="dataArray">
      <dataArray dimension="2"/>
      <SpatialCoverage resolution="1">
         <Latitude North="90" South="-90"/>
         <Longitude East="180" West="-180"/>
      </SpatialCoverage>
      <TemporalCoverage>
         <Date start="1986-01" end="1986-01"/>
      </TemporalCoverage>
   </SpatialModelTE>

</TEDataModelType>

</TE>
```

Figure 4.7 A snapshot of XTMS metadata. The figure shows the type of the dataset as a two dimensional "Array" along with the spatial and temporal coverage.

Chapter 5 . HIDE Architecture

5.1. Introduction

The HIDE architecture follows an extended wrapper/mediator approach called as plugin/ wrapper/mediator. The XML based wrappers handle the syntactic differences in the underlying data models, querying capabilities and low-level access level information. The structural and semantic differences are managed by the tree model. The mediation engine, upon receiving the query, finds the suitable sources (DataNodes), decomposes the query into appropriate low level queries, retrieves the data via multiple sources, transforms the data models to HIDE data model, combines the result and presents to the user. Most of the information systems assume a standardized mechanism to access the external data source such as SQL for a relational database. However such generalization cannot be applied in this scenario, as datasets of hydrology community are often complex and diverse. Hence an extended framework is defined for the wrapper/mediator architecture that provides a mechanism called as plugins, to extend the functionalities of the mediation engine.

Due to the heterogeneous nature of the data engines, we identified a layered approach is appropriate for handling various issues in data integration. The problems of integration can be divided into sub-problems and handled at each layer. Our system architecture is shown Figure 5.1

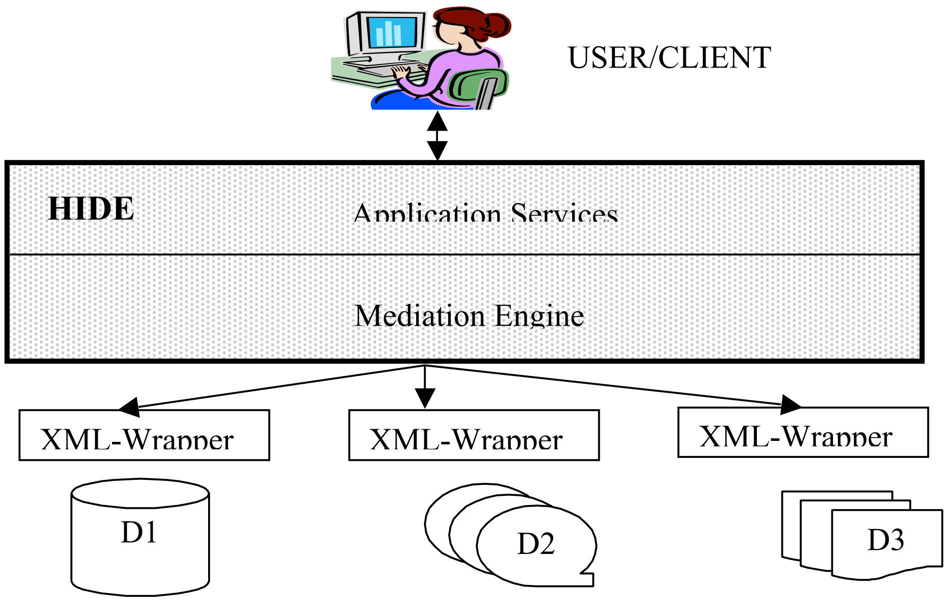


Figure 5.1 HIDE – A wrapper/mediator Architecture

The application layer incorporates the search engine, visualization tools etc. The mediation engine is layered as Data Model layer, transformation engine and Access Engine layers as shown in Figure 5.2. The metadata model (description, operational, transformation, access and syntax) acts as the XML wrappers for each individual data source. Apart from these, to extend the functionalities of each layer in the mediation engine with respect to a data source or a data concept, the architecture uses plugins as a mechanism to plugin the extensions.

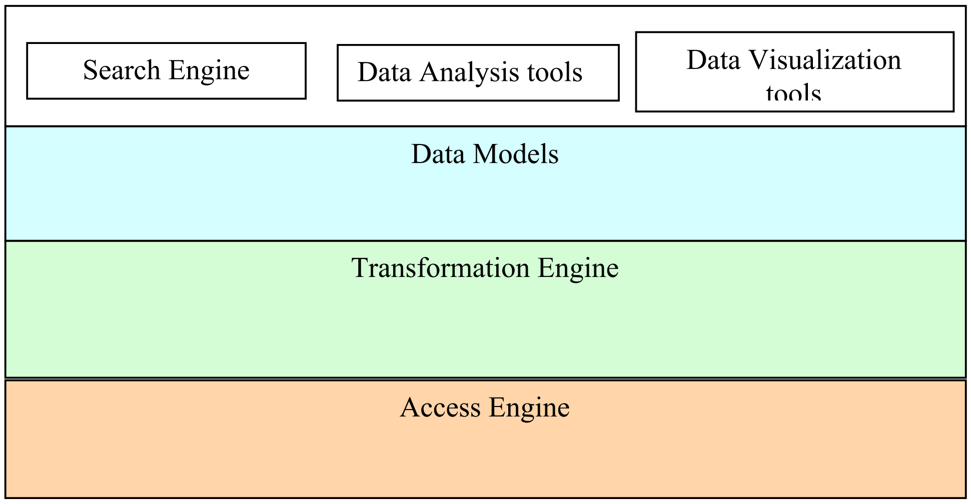


Figure 5.2 HIDE Architecture.

The HIDE prototype is implemented as a web-based Integration, data analysis and management system with functionality of a search engine, query, aggregation and visualization of data. The system uses client-server architecture, with Apache-Tomcat web server running at the back end. The HIDE server handles accessing external data engines, data querying, data modelling and transformation of the data. HIDE employs a thin client capable of user interface generation and handling of user interactions which interacts with the HIDE server. We use XML language for representing the metadata. Once all the metadata files are defined for a data engine, the system automatically performs the integration. Each layer in the HIDE architecture is developed as a Java package in the prototype. The interface between each layers are open through java "interface." This provides more flexibility and extensibility. Any additional features required in a layer can easily be plugged in without affecting other layers.

The prototype has integrated four different kinds of data sources and engines USGS [2004], GPCC [2001], Australian Antarctic Automatic Weather Station data engine [2004], and OFS data from NWS (National Weather service) with two levels of integration.

5.2. Design Objectives

The following principles drive the design of our architecture. The principles are derived from the fact that all data systems are autonomous, and work in a multi-user and heterogeneous environment where a single rule is not enough to define a diverse system. Our design objectives of the data integration architecture also incorporate the basic principles of extensibility, flexibility, and transparency.

5.2.1.Neutral Mechanisms:

Most of the scientific computing environments use file systems for storage of the data. The scientific community has defined various data formats for representing these complex temporal and spatial data. What scientists really want to do is to manage the data rather than the files. The Integration architecture of HIDE is completely independent of the low level mechanisms of the data storage, file organization and search methods. This goal is achieved by identifying each data engine as a separate entity with an interface to access the data. The entity encapsulates the complexity involved in the data storage and data search which is handled by the data engine based on the existing scientific standards and is separate from the Integration architecture.

5.2.2.Uniform Information Infrastructure:

One of the key issues of an integration environment is to define a uniform information infrastructure to the users. This enables an easy integration of data engines and, a uniform query pattern. The application services should not be aware of the specific low level mechanisms required to access data which are diverse geographically as well as in structure and content. Instead, applications are represented with a uniform view of data, with uniform interfaces for access across diverse and distributed data sources.

5.2.3.Layering Mechanism:

In our approach, the data integration problem is divided into multiple sub-problems. A layered architecture then is employed to solve each sub-problem. The inputs to these layers are templates which describes "how to integrate an external system to the layer." By using the layering approach, the technical problems of integration are confined within each layer and are simplified.

5.2.4.Flexibility and Extensibility:

Two fundamental principles of software systems are flexibility and extensibility. As the time grows, increasingly, new data systems with heterogeneity in data model, geographic location, accessibility can be identified for data integration and analysis. New standards for defining the data would emerge. Flexibility and ease in integration of these emerging data systems can also be achieved through the layered approach. The responsibility of each layer is defined through templates. The lower layers of the system require the understanding of the data and are defined through templates which contains the syntax and semantics of the data. Hence newer data standards can be plugged in through templates or as external packages into the system.

5.2.5.Simplicity:

We have kept simplicity in mind in the design and implementation of our architecture. We assume that data (sources/real datasets) are distributed across a network. The organization of all these data sources into a meaningful representation leads us to a simple tree based model where each node is defined as a data source or a dataset with defined relationships between them. The hierarchical methods of representation provide the necessary advantages of simplicity and extensibility. It is much easier to decide where to integrate a data source in a tree-based model rather than a flat model.

5.3. Object Factory

HIDE architecture employs the concepts of object factory methodology for the creation of various objects at each layer. Depending on the nature and corresponding identifier for the objects, the object factory methods create the necessary objects. This helps to separate the creation and management of the objects.

The Object factory methodology is being used at each corresponding layer of HIDE with exception on the application services layer. At Access engine layer, based on the type of accessibility defined through the XAD metadata file, the Access Engine factory creates the corresponding Access engines. For example, if the accessibility is defined as “http”, the Access engine factory creates a “HttpAccessEngine” object which can specifically handle the requirements of HTTP access. Similar factory methods are also employed at the data model layer and data model transformation engine layer. An object level representation of the object factory and objects are as shown Figure 5.3

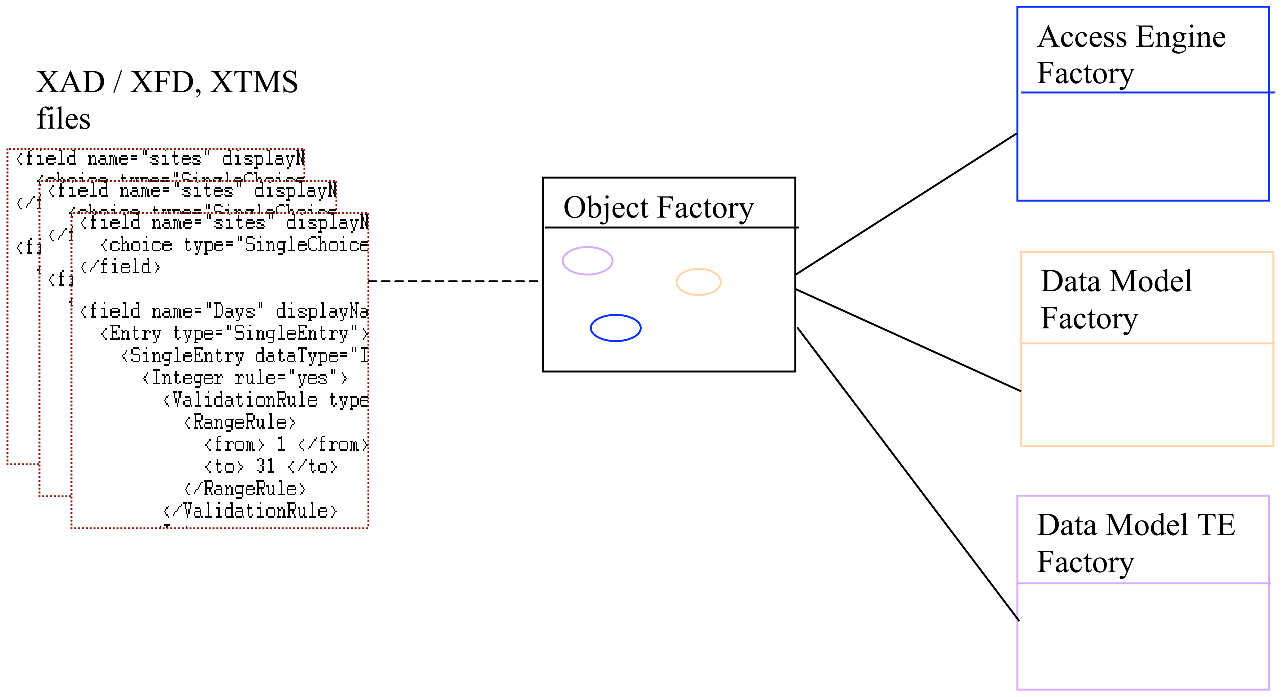


Figure 5.3 An object level representation of the object factory at each level and its relationships with the metadata files.

5.4. Application services

In this layer, various kinds of application services can be defined. These application services includes data visualization techniques such as 2D plots and 3D plots, data analysis methods such as query, multi-streaming of data, and a high level search engine. As the interface between each layer is open, new application services can be identified and added in without significant changes in the subsequent layers. This provides a great deal of extensibility. The application services use description model metadata and operational metadata for identifying the DataNodes and their query operations.

5.4.1.Search and Query Engine

One of the essential operations an earth scientist would like perform is to query multiple data sources / datasets uniformly, extract the data and perform various analysis. To facilitate this, we characterize operation – query at various layers of the system architecture.

Consider the following query, “What is the precipitation for a period of 2 days at the site Uchee creek in the state of Alabama”? There are two levels of query identified here. In the first level, the system has to identify the appropriate top-level DataNode to which the query has to be performed with the help of search engine. This type of query is identified in HIDE as a “search.” HIDE employs a search engine similar to “Google” for searching the appropriate DataNodes based on the query string provided by the user. The search engine uses the indices presented through the description model metadata for each DataNode in the DataNode repository to match against the “keywords” supplied by the user. To satisfy the above query, user has to find the appropriate data node by describing a search string “USGS real time precipitation from Alabama.” In this scenario, query is traced to the DataNode “Alabama” from the root based on the indices of precipitation, water data and Alabama as shown figure 5.4. The search engine then lists all the possible integrated data nodes. The user can view a subset of information of the data node along with a link to view the query interface to perform the query, in the list.

Once a DataNode is identified, the second level query is performed on the DataNode based on the query operational model defined in the DataNode. This kind of query operation is identified as a “data query.”
By contrast, consider another query from the data analyst, “what is the precipitation for the state of Alabama for 2 months including current month.” In this case, system identifies water as the appropriate node where the data query can be performed.

Consider another query, “what is the precipitation for a range of latitudes-longitudes for the period 1986-1987.” In this case, the precipitation node is identified by the system to perform query operations. It has to be noted here that this data query operations involves two different external sources. In this kind of data query of higher level DataNodes, the system translates the query to multiple queries, distributes them to multiple data sources. The data from each data source is then extracted and transformed into internal data models. Finally all the data from the multiple data sources are aggregated.

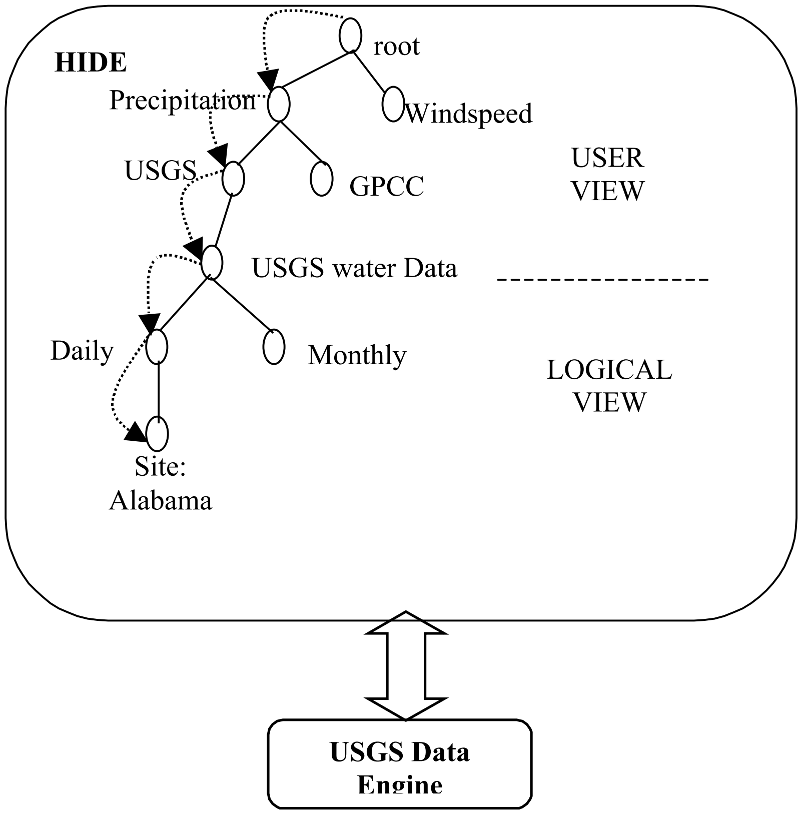
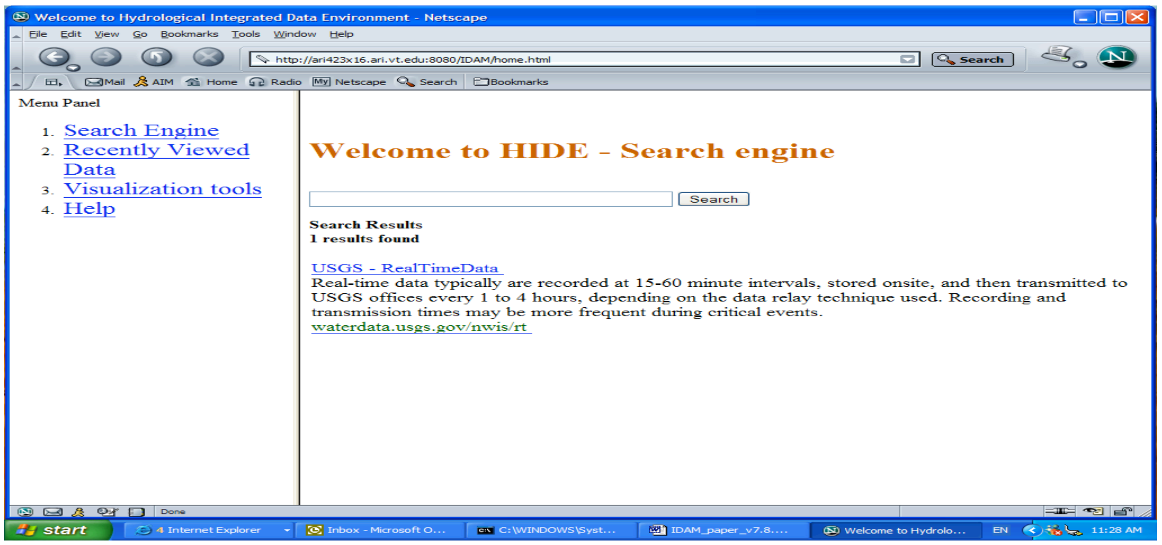


Figure 5.4 The trace for a simple query "precipitation for a period of 2 days at the site Uchee Creek in the state of Alabama.

As mentioned earlier, HIDE would list all the possible data nodes that satisfy the search string as shown in Figure 5.5. On selecting one of them, HIDE takes the user to the next level of search – a query screen (data query).



Welcome to HIDE - Search engine

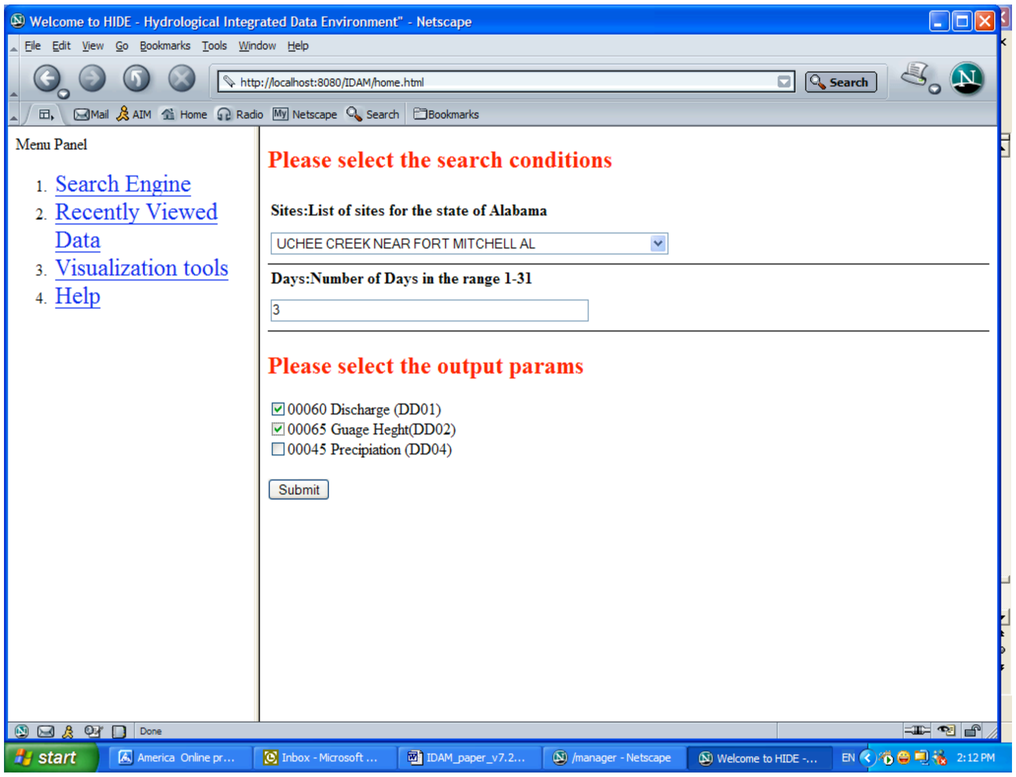
Search Results
1 results found

USGS - RealTimeData
Real-time data typically are recorded at 15-60 minute intervals, stored onsite, and then transmitted to USGS offices every 1 to 4 hours, depending on the data relay technique used. Recording and transmission times may be more frequent during critical events.
waterdata.usgs.gov/nwis/rt

Figure 5.5 Search Screen

5.4.2.Query Screens:

Our system provides a dynamically generated and uniform web based query interface to perform data query on a data node. The query interface is generated from the operational model (XOMS) with the search conditions and result parameters. The generation of query screens from the XOMS metadata files, validation of user entered parameters and the necessary changes is realized by the "operation" package of the HIDE system. A dataField object of necessary type (Entry, choice, Range) is used to define each search conditions and results. The validation rules for each condition is applied on to the data field. The operation model package creates the data fields and validation rules based on the XOMS files for each DataNode. Assume that a data analyst would like to perform a query "what is the discharge and gauge height from the real time data for the site UCHEE CREEK in the state of Alabama for a period of 3 days." The query screen for this query is shown in the Figure 5.6


Welcome to HIDE - Hydrological Integrated Data Environment" - Netscape
File Edit View Go Bookmarks Tools Window Help
http://localhost:8080/IDAM/home.html
Search
Mail AIM Home Radio Netscape Search Bookmarks
Menu Panel
1. Search Engine
2. Recently Viewed Data
3. Visualization tools
4. Help
Submit
Done
start

Please select the search conditions

Sites:List of sites for the state of Alabama

UCHEE CREEK NEAR FORT MITCHELL AL

Days:Number of Days in the range 1-31

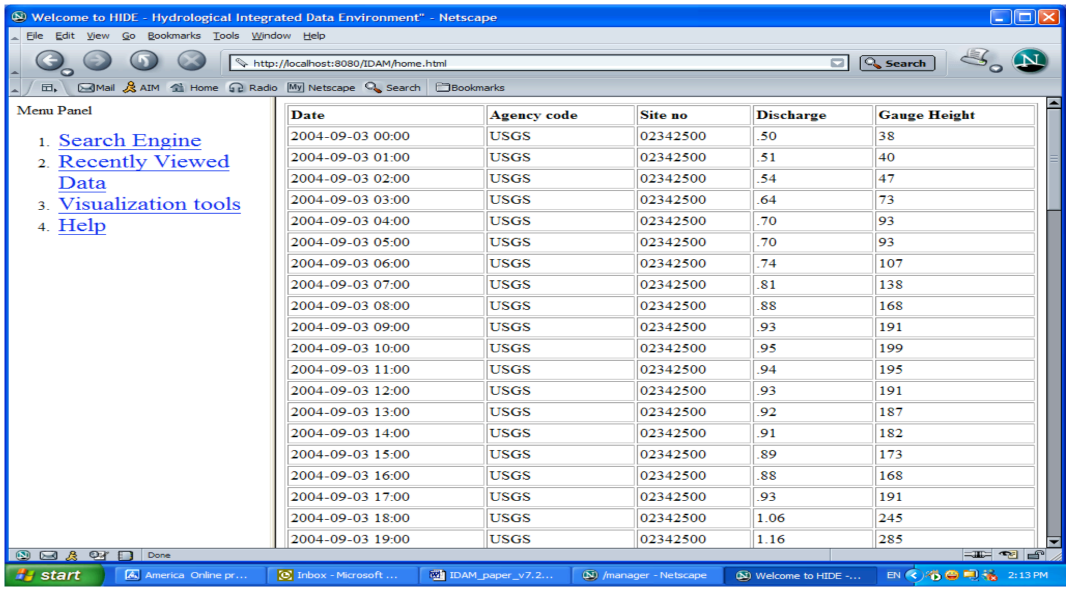
3

Please select the output params

☑ 00060 Discharge (DD01)
☑ 00065 Guage Heght(DD02)
☐ 00045 Precipiation (DD04)

Figure 5.6 Query Screen

Considering the above mentioned query, user can select the appropriate site "Uchee creek", a range of 2 days and result parameter as "Discharge and Gauge Height." Once the user has submitted the query, the data is retrieved, transformed and can be viewed as shown Figure 5.7

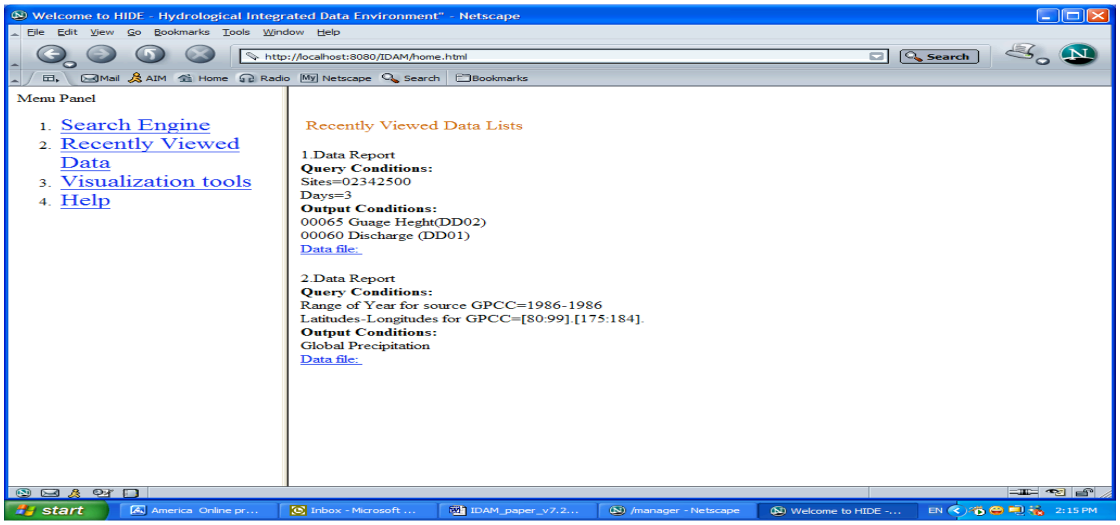


Date	Agency code	Site no	Discharge	Gauge Height
2004-09-03 00:00	USGS	02342500	.50	38
2004-09-03 01:00	USGS	02342500	.51	40
2004-09-03 02:00	USGS	02342500	.54	47
2004-09-03 03:00	USGS	02342500	.64	73
2004-09-03 04:00	USGS	02342500	.70	93
2004-09-03 05:00	USGS	02342500	.70	93
2004-09-03 06:00	USGS	02342500	.74	107
2004-09-03 07:00	USGS	02342500	.81	138
2004-09-03 08:00	USGS	02342500	.88	168
2004-09-03 09:00	USGS	02342500	.93	191
2004-09-03 10:00	USGS	02342500	.95	199
2004-09-03 11:00	USGS	02342500	.94	195
2004-09-03 12:00	USGS	02342500	.93	191
2004-09-03 13:00	USGS	02342500	.92	187
2004-09-03 14:00	USGS	02342500	.91	182
2004-09-03 15:00	USGS	02342500	.89	173
2004-09-03 16:00	USGS	02342500	.88	168
2004-09-03 17:00	USGS	02342500	.93	191
2004-09-03 18:00	USGS	02342500	1.06	245
2004-09-03 19:00	USGS	02342500	1.16	285

Figure 5.7 DataView

5.4.3.Recently viewed Data

HIDE keeps track of all the data retrieved by the user for the current session using the recently viewed functionality. It will list all the data retrieved, along with its search conditions and output parameters, as shown in Figure 5.8. This functionality facilitates the further data aggregation and fusion from those retrieved datasets as needed.


Welcome to HIDE - Hydrological Integrated Data Environment" - Netscape
File Edit View Go Bookmarks Tools Window Help
http://localhost:8080/IDAM/home.html
Search
Mail AIM Home Radio Netscape Search Bookmarks
start

Menu Panel

1. Search Engine
2. Recently Viewed Data
3. Visualization tools
4. Help

Recently Viewed Data Lists

1.Data Report
Query Conditions:
Sites=02342500
Days=3
Output Conditions:
00065 Guage Heght(DD02)
00060 Discharge (DD01)
Data file:

2.Data Report
Query Conditions:
Range of Year for source GPCC=1986-1986
Latitudes-Longitudes for GPCC=[80:99].[175:184].
Output Conditions:
Global Precipitation
Data file:

Figure 5.8 Recently viewed Data List

For every data retrieved by the user for the current session, the HIDE server notes the data file corresponding to the data retrieved along with the a report of the data in a “recently viewed report” file. Hence when user requests his/her recently viewed data, the HIDE server displays the information from the corresponding “recently reviewed report” file for the user.

5.5. Data Models

A model is an abstract representation of a real-world process. In the hydrology domain, the data sets are usually characterized by the nature of the data with respect to time and space. Based on these special characteristics of the hydrology data (temporal and spatial), the HIDE architecture uses a simple temporal and spatial model for representing all the data retrieved from multitude of data engines irrespective of their internal data model. Hence this layer provides a uniform representation of the data, while masking the lower layer details. The temporal model models temporal data while the spatial model models spatial data. A combination temporal-spatial model combines both kinds of data.

5.5.1.Mathematical Analysis of models

Temporal Model: - Time specific data often arises when monitoring real world events or measurements such as precipitation or temperature. The data can be univariate or multivariate and can be continuous or discrete. A univariate data model consists of single (scalar) observations recorded over time intervals, while a multivariate model consists of multiple measurements or observations recorded over specific time intervals. Temporal models are often used in forecasting and statistical analysis.

HIDE assumes that data are equi-spaced and multivariate in nature. A sample temporal dataset is shown in figure 5.9.

```
Source   StationId        DateTime   Discharge Precipitation
USGS    02342500 2004-11-13 00:00       130     .00
USGS    02342500 2004-11-13 01:00       125     .00
USGS    02342500 2004-11-13 02:00       125     .00
USGS    02342500 2004-11-13 03:00       130     .00
USGS    02342500 2004-11-13 04:00       125     .00
USGS    02342500 2004-11-13 05:00       130     .00
USGS    02342500 2004-11-13 06:00       130     .00
USGS    02342500 2004-11-13 07:00       142     .02
USGS    02342500 2004-11-13 08:00       154     .00
```

Figure 5.9 A Sample Dataset for temporal data

HIDE also assumes that data consists of stationary attributes (Source, StationId in figure 5.9) and non-stationary attributes which varies with time (Discharge and Precipitation in figure 5.9). The mathematical temporal model in HIDE can be represented as follows.

Assumptions:

1. The data is multivariate in nature.
2. The data can be equally spaced in terms of time. This condition holds for data aggregated from multiple data source, where each data sources, can use different time steps.
3. The date and time are represented in 12z time zone.

Let $Z(t)$ represents the temporal data and $X(t)$ represents the non stationary temporal data and Y represents the stationary data.

$X(t) = \{ x_1(t), x_2(t) \ldots\ldots x_n(t) \}$; n = number of non-stationary attributes

$Y = \{y_1, y_2, \ldots\ldots y_m\}$; m= number of stationary attributes

$Z(t) = X(t) \cup Y; \forall t$

And $t_r = t_{r-1} + s$; s = time step.

Spatial Model: The aspects relevant to the geographical nature of data are spatial data (where the data is present) and attribute data (what is present). With respect to the computer models, Geographic Information Systems (GIS) identify two spatial models vector model and raster model. The Vector model is often used for representing objects such as rivers, bridges etc and the Raster model is usually used for representing attributes such as temperature, elevation, etc., which are off importance to the hydrology community. HIDE models the spatial data as a raster model.

A vector model characterizes data with points, lines and regions (polygons). The points are defined using Cartesian coordinates (x and y). Lines and Regions are defined by a series of ordered points.

A raster model represents data as a matrix or grid of cells. Every cell has a unique reference coordinate at the top left corner of the cell or centroid of the cell. For a resolution of r degrees, and a known latitude, longitude (la,lo) of the top left corner of the cell, a cell can be represented as, (la,lo) , (la,lo+r) , (la+r,lo), (la+r,lo+r). Every cell has an associated attribute attached. Multiple attributes of the data can be modeled as individual grids of values which are then overlaid on a single grid as shown in the figure 5.10.

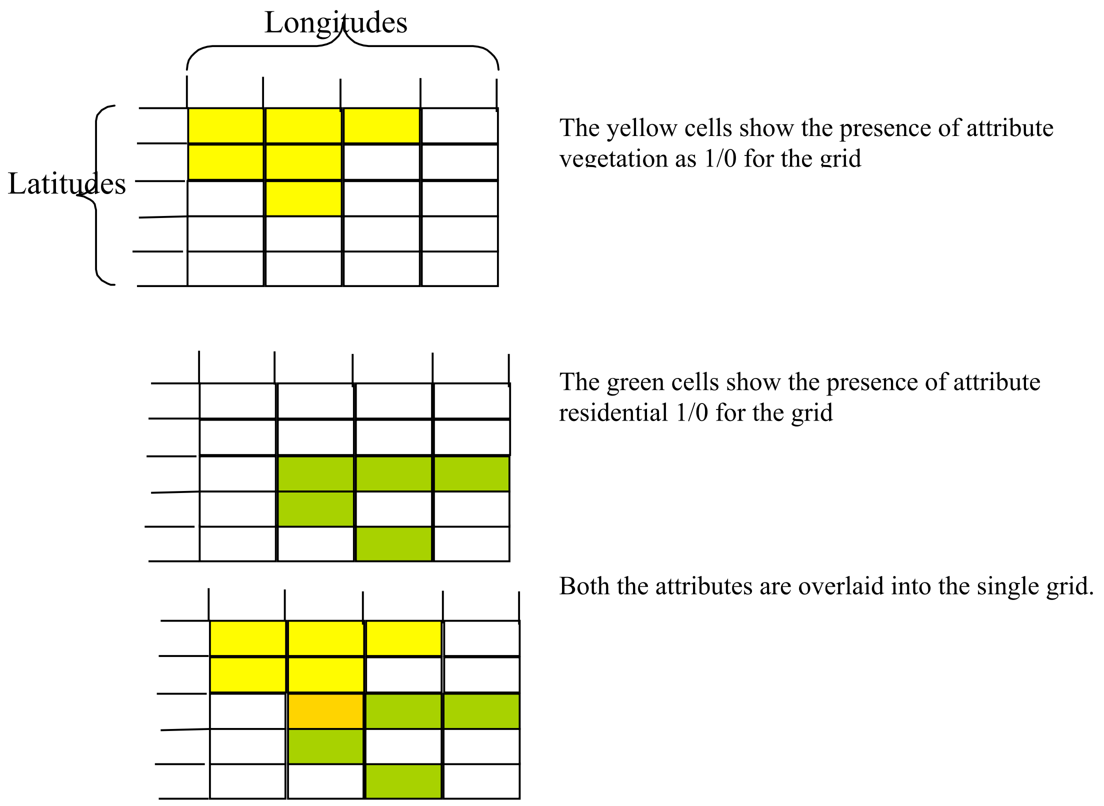

Longitudes
Latitudes
The yellow cells show the presence of attribute vegetation as 1/0 for the grid
The green cells show the presence of attribute residential 1/0 for the grid
Both the attributes are overlaid into the single grid.

Figure 5.10 Raster model representation.

Based on the continuous nature of the data, HIDE models data as a raster model and can be represented mathematically as follows.

Assumptions:

1. The spatial parameter is represented as absolute coordinates of latitudes and longitudes of the top left corner of the cell.
2. The cell size is uniform for the entire grid. This holds true even when the data is aggregated from multiple data sources.

Let P represents a set of latitudes for a specific range and Q represents a set of longitudes for a specific range.

Let m = number of latitudes for the grid.

n = number of longitudes for the grid.

$P = \{ p_1, p_2, \ldots p_m \}$

$Q = \{ q_1, q_2, \ldots . q_n \}$

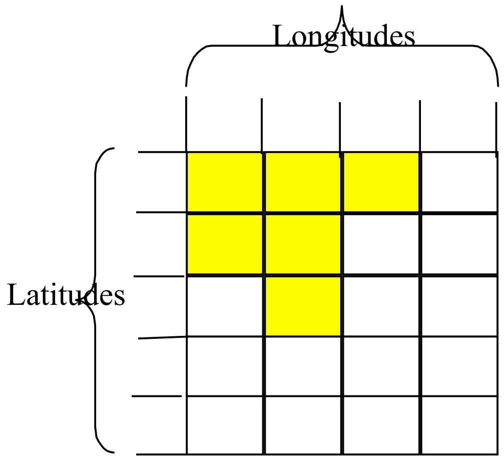
$S = P \times Q$ (A Cartesian product of P and Q)

$X(s)$ = set of attributes for cell s

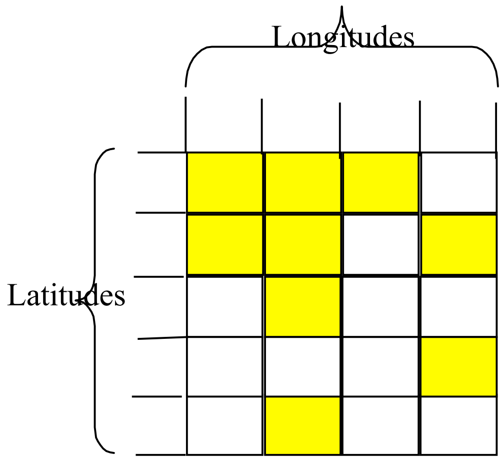
$= \{ x_1(s), x_2(s), \ldots . \mid s \in S \}$

Temporal-Spatial Model

The temporal-spatial model is a combinational model, for modelling data with temporal and spatial characteristics. HIDE models the spatial nature of the data as a raster model. This model is particularly useful when the measurements are recorded for a region of space at different time instants. This is shown in figure 5.11.



The shaded cells show an attribute measurement at time t.



The shaded cells show an attribute measurement $t+t_s$ (t_s = time step)

Figure 5.11 A temporal-spatial model for a univariate data.

The model can be mathematically represented as the following.

Assumptions:

1. Data is univariate in nature.
2. Data is equally spaced with respect to time.

Let P represents a set of latitudes for a specific range and Q represents a set of longitudes for a specific range and X is an attribute measurement.

Let m = number of latitudes for the grid.

n = number of longitudes for the grid.

$P = \{ p_1, p_2, \ldots p_m \}$

$Q = \{ q_1, q_2, \ldots . q_n \}$

$S = P \times Q$ (A Cartesian product of P and Q)

$X = f(t,s);\ \forall t, \forall s$

5.5.2.Object Models

HIDE realizes, every dataset retrieved from a data engine as a Data object. The Data objects in HIDE are associated with a Data model object which essentially comprises of mathematical model (temporal / spatial), a data information model and a syntactic representational model. The data information model in HIDE necessarily captures the descriptive information about the data such as the list of data sources from which the data is retrieved. This information is particularly relevant when the aggregated or retrieved data from HIDE is to be exchanged with other data sources. The syntactical model captures the representational details of the data.

The data models objects in HIDE is implemented as the data model package. The basic nature of the data, i.e. information model and representational model is encapsulated in the base class of "DataModel" while the temporal and spatial characteristics (mathematical model) of the data is realized by the subclasses Spatial , temporal and temporal-spatial model.

The data model object contains following information.

1. Information model

   a) List of data nodes (data sources) from which the data has been retrieved and modeled

   b) The temporal and spatial coverage for the data if any. The temporal coverage can be a single data-time point or a range of data-time points. The spatial coverage is represented as a bounding rectangle coordinates. The limits of coverage of a data set are expressed by latitude and longitude values in the order western-most, eastern-most, northern-most, and southern-most. For data sets that include a complete band of latitude around the earth, the West Bounding Coordinate shall be assigned the value -180.0, and the East Bounding Coordinate shall be assigned the value 180.0 . If the bounding area is a single point, the same values are used for northBoundingCoordinate and southBoundingCoordinate, as well as for westBoundingCoordinate and eastBoundingCoordinate

2. Data representation model
    a) Type of the data (ASCII/binary)
    b) Format of the data (comma-seperated/tab-sperated).

The temporal characteristics of the data are reflected in the temporal data model object. The object follows the mathematical temporal model as described in the section 5.5.1 The information contained in the object includes,

1. List of Data fields
2. Starting date time value.
3. The date time format. HIDE uses ISO 8601 Date and Time Specification. An example for the time date format is YYYY-MM-DD
4. Time steps.

The spatial characteristic of the data is reflected in the spatial data model object. The object follows the mathematical spatial model as described in the section 5.5.1

The information in the object includes,

- List of Data fields
- Resolution of the cells in the grid.
- Range of latitudes
- Range of longitudes

The temporal-spatial data model objects combine both characteristics of temporal and spatial.

Even though the data model objects are being implemented as “objects”, there exists a necessity of the representation of the model objects as a “document” for the better understanding of data exchange with other systems. This is achieved by the representation of the models in a dataobjectmodel schema (See Appendix). Or in other words dataobjectmodel schema acts as a template for the data. It provides the information about the mathematical models, data source information model and syntactic representation model. The schema acts as an abstract representation of the DataModel object and is equivalent to the “DataModel” class. Hence a concrete implementation of the schema which are the DataModel metadata, is equivalent to DataModel objects in the system. Therefore for every data object, there exists a DataModel object and a DataModel metadata. These relationships can be shown as in figure 5.12

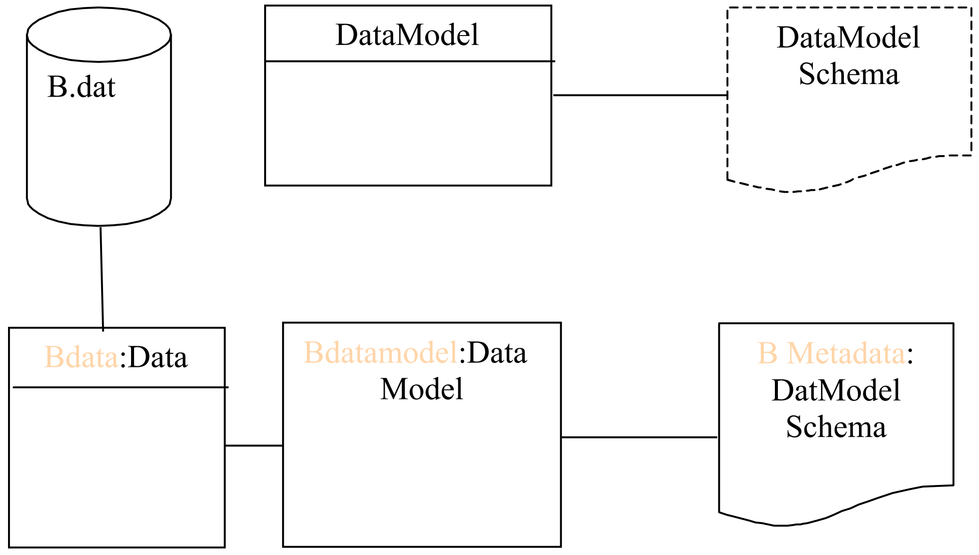


Figure 5.12 The relationship between the data, dataobjects, DataModel and DataModel schema for a dataset B.dat

5.5.3.Aggregation Model

The DataNode tree of the HIDE system enables the user to retrieve the data sets from multiple data engines based on the query provided, and aggregate the data. Each dataset retrieved is transformed to a data model object (temporal or spatial) and then aggregated using the aggregation model object. The aggregation is done based on the nature of the data. If the all the datasets retrieved have a temporal data model, the aggregation is done with the relevance to the temporal nature of the data. The similar rule applies to spatial data model as well. If some of the datasets are temporal in nature while some has spatial in nature, the aggregation data model, aggregates the data based on both.

For example, Assume that, user retrieves data from data source A for a time period of 06/06/2001 – 06/12/2001 and data source B with a time period of 07/1/2001 – 07/12/2001. The system retrieves the datasets from source A and source B, transforms each into a temporal data model with the above specified range and aggregate using the aggregate data model. The resulting DataModel after aggregation would be temporal in nature, with a range of 06/06/2001 – 07/12/2001.

It has to be noted here, that the data is being aggregated and not accumulated. Hence aggregation data model cannot tolerate any overlap in the temporal and spatial characteristics of the data.

5.6. Transformation Engine

The complex nature of scientific datasets can be defined using various standards such as in DODS [2004] for oceanography data. Each of these standards has a defined data model, for example; DODS has defined data types of 'Structure', 'Sequence', 'Arrays' and 'Grids' for modeling various kinds of data. Each data engine uses multiple ways for modeling the data.

The transformation engine performs a transformation of the retrieved data irrespective of the underlying data model to the data model (temporal / spatial) of the HIDE. To achieve this, Transformation engine requires an understanding of the underlying data engine's data model, metadata and syntactic information of the data. The XFD metadata describes the data model and syntactic information of the data, while the XTMS metadata files provides the necessary information for the transformation such as data model to be converted to (temporal/spatial), the temporal and spatial coverage of the data and the temporal/spatial characteristics of the data.

The transformation engine factory creates the requisite transformation data model objects (temporal, spatial, and temporal-spatial) on the basis of the data nature provided by the XTMS metadata. The respective transformation engine objects reads the data with help of syntactic metadata (XFD) and transformation metadata (XTMS) , transforms the data into a corresponding DataModel and serializes the data model for later use. A schematic representation of the transformation engine is shown below.

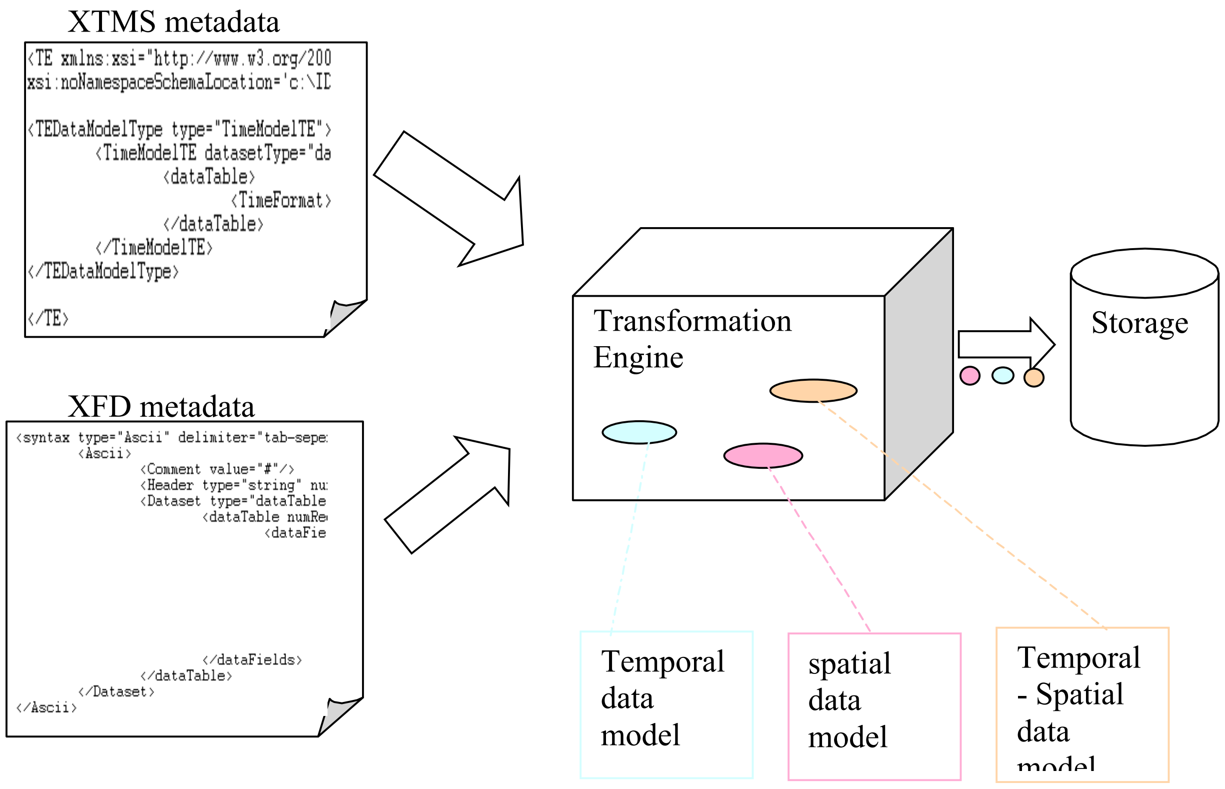


Figure 5.13 A schematic representation of Transformation engine and its operations

5.7. Access Engine

This layer realizes the access methods for connecting to the external data engine.
It identifies how to connect to the external data engine, what are the parameters to be supplied, management of the connections, the operations to be performed if any error occurs and retrieval of the datasets. It handles all the complexities of making a lower layer connection to the external engine. The access engine defines an access point with necessary information from access metadata (XAD).

The Access Engine layer is realized by the Access engine package. The Access engine factory creates different kinds of access engines (http, ftp) for each leaf DataNode based on the corresponding XAD metadata files. It has to be noted here that, usually access engine interacts only with leaf data nodes as they connect with the data engines. The Access Engine object keeps all the information required to make an external connection. The information includes type of connection and a list of parameters to be sent to the external data engine.

The HttpAccess Engine, handle and manage an http connection to the external data engine. When a user submits the query to the data node, the HttpAccessEngine object makes a connection based on the connection details provided by the XAD file, and sends the search parameter names along with the user entered values to the external data engine. On receiving the reply data, the engine retrieves the data and stores in a file and sends it back to the transformation engine to begin the transformation process. If any error occurred, the HttpAccessEngine object replies back with an error. Similarly an FTPAccess Engine, handle and manage the ftp connections with the external data sources.

5.8. Plugins

As explained in the above sections, each layer in the HIDE architecture has a defined role and responsibility in the system. For example; Access Engine handles the complexities of accessibility with the external data engines. Sometimes, it becomes necessary to extend the functionality of the layers for a specific data engine. Consider a data engine that requires a certain set of functionalities in any layer. This adds an extra responsibility to the layer, but is limited to a specific data engine. A solution to this problem is to extend the corresponding Engine in the object oriented paradigm. However the runtime environment of the HIDE system is limited to the data engines. To avoid these limitations, HIDE employs plugins at each layers of the system.

Plugins can be considered as “slots” through which new objects can be “plugged-in.” For example; the AccessEngine object may not cover all comprehensive methods of accessibility to the data entities. As new methods are being evolved, it becomes quite impossible to realize all possible ways of connecting to external data entities. Moreover some data engine might employ its own data access protocol or requires some data preparation before accessing them. Hence to plugin a new Access Engine object, the data sources can create the new Access engine as inherited from the AccessEngine of the HIDE system, and plug in through the Plugin for the AccessEngine layer. Any external access engine object can be plugged into the system without affecting the runtime nature of the system. The Plugin for the Access Engine acts as a point of contact for plugging into the AccessEngine layer of the system. Once the external access engine object is plugged in, the object can be used like any other access engine objects. All the information required for plugin is supplied through the XAD files. The use of plugin access engine object assist to separate the access engine object of the HIDE system and the external system facilitating the encapsulation of any necessary security mechanisms required for the external access, specific to the data engine in the external access engine object. It is assumed here that external access engine follows the rules of the AccessEngine object and can be plugged into the system.

A schematic representation of the plugin for the Access Engine layer is shown in figure 5.14

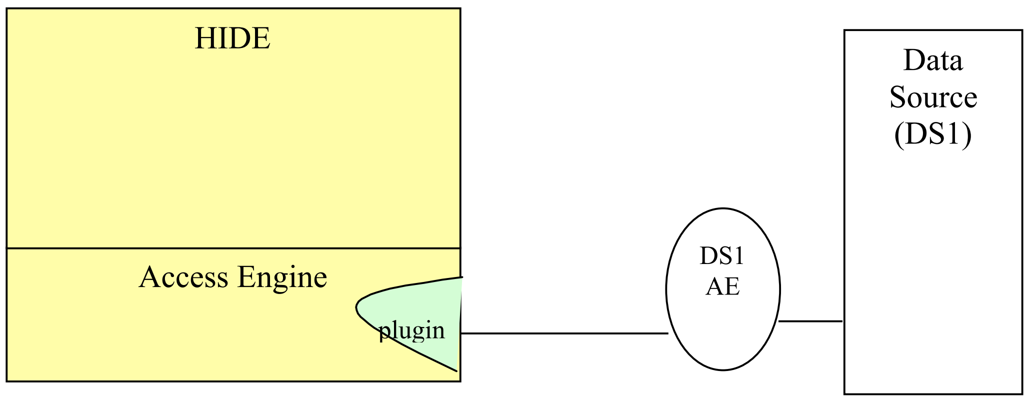


Figure 5.14 A schematic representation of the plugin for the Access Engine layer.

In the figure, assume that Data Source DS1, require to extend the functionality of the Access Engine layer while maintaining the runtime environment of the HIDE system. The extended AccessEngine for DS, DSAE1, is plugged in to the AccessEngine layer of HIDE through a Plugin. The information about the plugin is provided through XAD metadata for the data engine. Similar methods can be employed to other layers of the system as well.

5.9. Integration Levels

One of the characteristics that distinguish HIDE is the capability to provide different levels of integration required for different data sources and users. We have identified two levels of integration in our system, primary level and secondary level.

The primary level integration enables a complete integration of the data engine to HIDE system. This includes searching, querying the data and performing the data analysis and visualization. The secondary level integration enables only partial integration of the data engine to HIDE system. This includes searching and querying the data without any analysis and visualization features of the HIDE system.

Defining these integration levels provides multiple uses. One such kind of use would be, if we would like to integrate a highly complex system but could not be completely defined in HIDE system, the secondary level integration would be an ideal choice. Another example is, if we would like to integrate a system but due to security reasons would like to limit the degree of integration, the secondary level would be again an ideal choice.

Chapter 6 Data source(s)/ Dataset (s) Registration

HIDE architecture considers scalability and flexibility as fundamental design objectives. But the primary concern for any data integration mechanisms is simplicity. A simple data source/dataset registration mechanism is the first and foremost requirement for any data integration mechanisms. We have defined the registration methodology to HIDE as a four step process. It involves the following,

1. Which degree of integration you require? For a complete integration, primary integration is an ideal choice. If the data entities have specific concerns, then to address that, choose secondary.
2. Do you want to integrate at the domain level (applicable for domain analysts) or integrate at the logical level (Applicable to logical analysts) and to which DataNode?
3. Is the DataNode sub tree available for the data source/ dataset? The creation of sub tree requires the development of the metadata model (XDMS, XQMS, XAD, XFD, XTMS) for each DataNode and identification of the hierarchical representation of it. At the domain level, the creation of DataNode sub tree should be based on the domain hierarchy or the taxonomical relationships between the concepts realized by the DataNode. At the logical level creation of sub tree should be based on the structural organization of the data.
4. Integrate the DataNode sub tree by making changes in the parent DataNode to which the sub tree is integrated.

Following sections would illustrate a more comprehensive description on the registration process with an example.

6.1. Registration with primary level Integration

The HIDE architecture enables a system integrator to add a new data source with little effort. It involves only the addition of new metadata files.

Assume that a system integrator integrates a new data source such as “TOVS-Pathfinder Atmospheric data” to HIDE. The integration of the new data source can be summarized in the following steps.

1. Identify and create a DataNode sub-tree in the logical view.
2. Define the operational model of each DataNode in the DataNode sub-tree.
3. Define the access point for each leaf DataNode in the DataNode sub-tree.
4. Define the transformation information for each leaf DataNode in the DataNode sub-tree
5. Define the syntax and semantic information for each leaf DataNode in the data node sub-tree.
6. Add the DataNode sub-tree of the data source to the DataNode tree of the HIDE system.

Each of these steps is described in the following paragraphs.

The process of integration begins with the identification of the desired integration levels. As the above mentioned data source can be completely recognized by the HIDE, the integrator can choose the primary level of integration. This information is recorded in the operational model metadata files.

The first step involves identification and creation of the DataNode tree in the logical view. In the above mentioned data source, a DataNode could represent the data sources with leaf nodes depicting the years and months for which the data is available as shown Figure 6.1. It has to be noted here that the logical view and physical of a DataNode sub-tree can be identical.

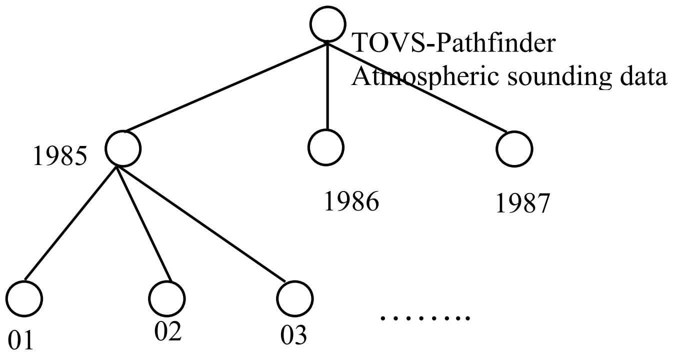


Figure 6.1 A DataNode sub-tree for the data source TOVS.

The creation of data node tree is achieved by defining the various metadata. Each data node in the tree is described using XDMS specification file. This metadata includes information such as identification, labels, indexes (for data search operations), additional information such as history, URL, the operations supported and list of its child DataNodes. This step is repeated until all the DataNodes in the DataNode sub-tree is described.

Definition of operational model begins with the identification of the model for each data node. In the above mentioned scenario, as the data source supports only query operations, a query operational model can be defined for the data nodes. It has to be noted here, that it is not mandatory to define the model for all the data nodes except for the leaf nodes. Hence in the higher levels of data node hierarchy, the query model describes various aspects of the query operations and aggregation of the data, while lower models would also define the access point information, transformation information etc. The query model for a data node is specified through the XOMS files. In these, integrators can specify the query conditions fields, the validation rules for each field and the output result fields.

The third step involves the definition of access point information for the leaf DataNodes. This provides the leaf data nodes with information about "how to access the external data source." The integrators can specify the protocol to be used, and the query string format to extract the data.

In the fourth step, integrators define the transformation model for transforming the data extracted from the data source. This information can be specified through the XTMS files. In these files, the integrators can provide the data model (spatial/temporal), spatial and temporal coverage of the data from each data node.

The definition of syntax and semantic information for the data extracted from the DataNode can be specified through XFD files. This information would be used by the transformation engine to extract the data, analyze and transform.

In the final step, the integrator identifies the appropriate data node to which the sub-tree belongs and which can acts as its parent DataNode. Then the sub-tree information can be specified in XDMS specification of the parent DataNode. It also has to be noted here, that if the parent DataNode has a query model already defined, then necessary changes are to be made in the query model to include the newly added data source. The final data node tree is shown in Figure 6.2. In the above mentioned scenario, as the data source provides the precipitation data, the DataNode sub-tree is added into the precipitation DataNode.

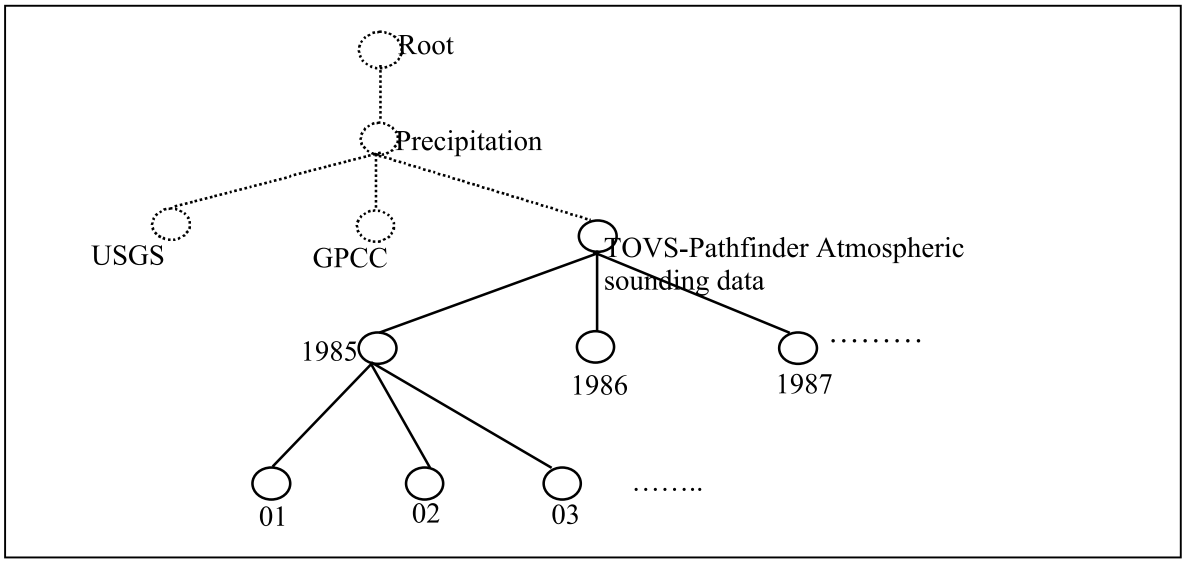


Figure 6.2 The DataNode tree after adding the new data source.

The process of addition of a new dataset follows similar steps as mentioned above. But, a new dataset is identified as a single DataNode. Hence the steps are repeated for a single DataNode.

6.2. Registration with secondary level of integration

The secondary level of integration enables the system integrator, to integrate those data systems which cannot be completely recognized by the HIDE architecture. The process of such kind of integration can be summarized in the following steps.

1. Identify and create the DataNode sub-tree in the logical view.
2. Identify the operational model.
3. Add the DataNode sub-tree of the data source to the DataNode tree of the HIDE system.

The process of integration begins with identification and creation of the DataNode sub-tree of the particular data source. Each DataNode in the sub-tree can be described through the XDMS files which include documentation, child DataNodes and supported measurements. This step is repeated until all the DataNodes are described.

The Definition of operational model begins with identification of the appropriate model of each DataNode in the DataNode sub-tree. The definition of the operational model is provided through the XOMS files. This would include the level of integration (secondary) and the link to the external operation interface. It has to be noted here that, as in this type of integration, HIDE system relies on the external data source interfaces to perform the operations; the data extracted cannot be aggregated from multiple data nodes, analyzed and transformed.

The final step involves the identification of appropriate parent DataNode in the HIDE DataNode tree to which the DataNode sub-tree belongs. Once the parent node is identified, the appropriate information of the DataNode sub-tree can be specified in the XDMS files of the parent DataNode.

Chapter 7 . NWS Dataset integration

7.1. Introduction

Operational research plays a critical role in the operationally hydrologic forecast mission at the NOAA National Weather Service (NOAA NWS) River Forecast Centers (RFCs) which is uniquely mandated amongst Federal agencies to provide forecasts for the Nation's rivers by providing daily and other forecasts at over 4,000 points across the contiguous United States and Alaska. The operational component of the mission is performed at 13 River Forecast Centers and approximately 120 Weather Forecast Offices at strategic locations across the United States.

The NWS Office of Hydrologic Development supports the operational mission by developing, implementing, and maintaining forecast models which are calibrated for specific rivers and streams based on historical events. Current observations and historical information is used for conditioning and constraining the models. Hence, a delayed, inaccurate, inconsistent, incomplete or insufficient data which is used for calibration of the forecast models can cause significant problems in the forecast process and for the forecasters who operate them.

Lack of appropriate tools could hamper hydrological calibrations, research and development for improving the operational-related hydrological river forecast research. HIDE system offers a data integration solution that helps in data exploration, analysis as new heterogeneous data sources are being added. HIDE provides metadata architecture with an open systems structure with interoperability and modularity. HIDE can access multiple large volumes of heterogeneous data sources, with different structures and formats and can easily adopt changes over time because of its modularity and simplicity. HIDE also provides users with a uniform GUI and can merge multiple datasets and data sources. As new data analysis is being introduced, the layered architecture of HIDE enables for customization for specific requirements.

In the following sections, the integration and customization of HIDE with the time series files (FS5Files) of River Forecasting centers (RFC's) of NWS are being discussed. It can be shown that, with the introduction of the data sources, the changes required in HIDE to perform the integration is minimal.

7.2. FS5Files

The NWSRFS Operational Forecast system (OFS) is a continuous river forecasting system which provides the forecaster with information required for flood forecasting. The System stores observed and future point measurements such as temperature, precipitation and produces forecasted products. These measurements are kept in specialized files known as FS5Files. For better development of forecast models, it becomes a requirement to integrate the time series information from these complicated FS5Files with the data sets from other RFC's, organizations such as USGS etc. FS5Files are a collection of files used by various components of the NWSRFC modules.

The processed data base files (PRD) of FS5Files are time series files which contain the time series data produced by preprocessor functions or forecast components. The preprocessor functions of NWSRFC write these time series data which were created for each RFC station or area, while the forecast component reads these time series and writes it back to the processed data base. The PRD time series files are basically collection of files with information such as various datatypes (MAP, MAT, and MPAE) and the measurements for a time instant. The time series data time intervals can be 1, 2, 3, 4, 6, 8,12 or 24 hrs. One time series information is a combination of files PRDINDEX, PRDPARM and PRDTS.

File PRDINDEX contains the index to the time series in the Processed Data Base. A hashing algorithm is used to determine the location in the index, based on the time series identifier and data type. The key is the time series identifier and data type. It consists of 16 byte binary records with each record containing the time series identifier, datatype code, and the record number of the first logical record in the time series file.

The PRDPARM file contains PRD control information and the datatype index. The file starts with a control information record and followed by a series of datatype records as shown in figure 7.1

PRD Control information
User identifier
Maximum number of datatypes
Maximum number of time series
Min # of days for the observed data to be kept
Actual number of time series
Actual number of datatypes
PRD Datatype Index
Datatype code
Unit number
of time series for this datatype
Maximum days data
Smallest time interval allowed
Processing indicator
Future data indicator
First record in the file of this type
Last record in the file of this type
Unit dimension
Number of values per time interval
Number of time series defined for a data type

Figure 7.1 A schematic representation of PRDPARM files.

The PRDTSn files contain the PRD time series data. They consist of 64 byte binary records with control and time series records. Each time series record consists of a header and time series information. An explanation for each record is shown Figure 7.2.

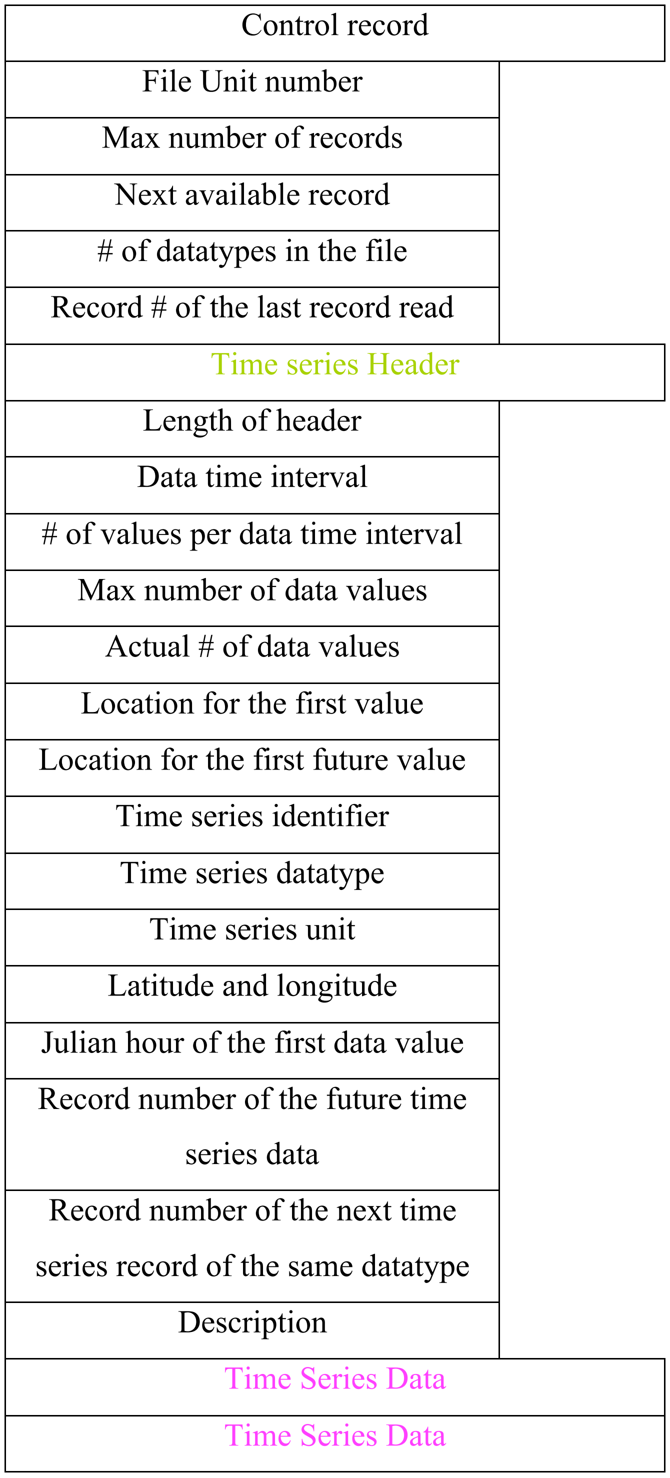

Control record
File Unit number
Max number of records
Next available record
# of datatypes in the file
Record # of the last record read
Time series Header
Length of header
Data time interval
# of values per data time interval
Max number of data values
Actual # of data values
Location for the first value
Location for the first future value
Time series identifier
Time series datatype
Time series unit
Latitude and longitude
Julian hour of the first data value
Record number of the future time series data
Record number of the next time series record of the same datatype
Description
Time Series Data
Time Series Data

Figure 7.2 A schematic representation of PRDTs files.

The time series files are being generated with 45 days worth data with 30 days of observed data and 15 days of forecasted data, collected and saved. All the FS5files are then retrieved, zipped and stored in the repository. This process occurs on a timely basis as part of the NWSRFC system. The zipped file contains all the FS5files along with the PRDINDEX, PRDPARM and PRDTS file. The name of the zipped file follows the pattern of “fs5filesyyyymmdd.” For example; if the FS5files are being saved on the day Nov 10th 2004, the name of the zipped file would be represented as “fs5files20051110.” The time series data from this collection of FS5files might contain 30 days of observed data i.e. from Oct 10th 2004 – Nov 10th 2004 and 15 days of forecasted data i.e. from Nov 10th 2004 – Nov 15th 2004.

7.3. Integration of Time series data of FS5Files to HIDE

The integration to HIDE (See Chapter 6) can be done at each layer, depending upon the complexities involved. Following steps are performed while integrating to HIDE. At present, FS5Files from the Ohio River forecast center are being integrated to the HIDE system.

1. Creation of a DataNode subtree. The DataNodes are identified and organized as a DataNode subtree. The DataNode subtree includes the hierarchical organization among RFCs. This would help to integrate FS5files from other RFCs with ease.
2. Defining description model metadata for all the DataNodes.
3. Defining the query model (XOMS), syntax (XFD), access model (XAD) and transformation model for the end leaf nodes
4. Integrate the DataNode subtree to the corresponding parent node in the HIDE resulting in a DataNode subtree as shown in the figure 7.3

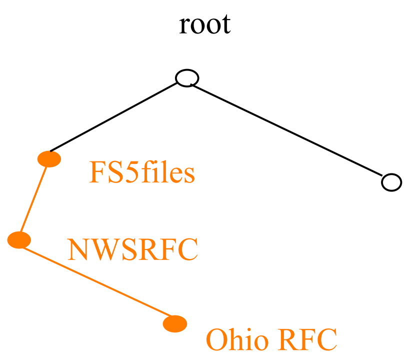


Figure 7.3 The DataNode tree for OFS data.

It is assumed that HIDE is part of the local network of the River Forecasting center. The integration of FS5files is performed with plugins and metadata files which act as wrappers for the OFS system. Due to the complex nature of the FS5files data and additional functionalities required at the Access Engine layer, the OFS data engine plugs-in its specialized OFSAccessEngine through the plugins of the HIDE system. A high level representation of the integration of OFS files to the HIDE is shown in Figure 7.4

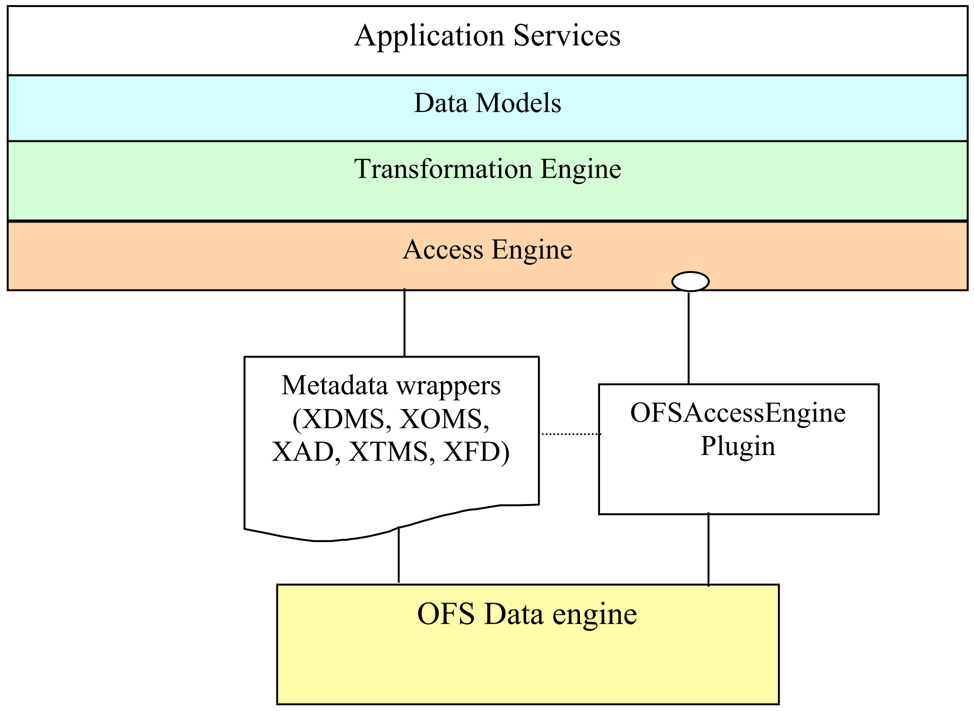


Figure 7.4 A high level representation of integration of OFS files to HIDE.

As explained in the previous sections, the integration to HIDE is performed at each layer and can be represented as in the Figure 7.4. Following sections briefly describes the integration process at each layer. As FS5Files are off temporal in nature, it perfectly fits with the temporal model of the HIDE system.

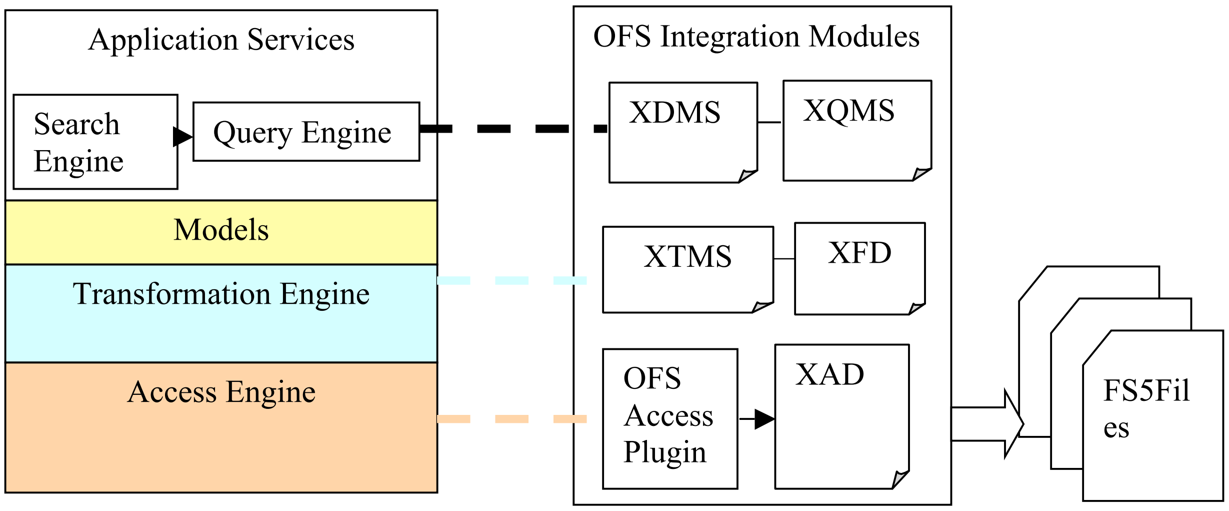


Figure 7.5 OFS Time Series Data Integration with HIDE

7.3.1.Integration to Access Engine Layer

The NWSRFC uses a utility "ReadOFS" to read the time series data from the FS5File for a particular saved date-time. The utility requires the information such as the date, time, datatype, time series identifier and time periods, reads the PRDINDEX, PRDPARM and PRDTSn files and retrieves the data that matches the input conditions. This utility acts as the interface for the FS5Files from HIDE and resides externally to HIDE.

As the FS5Files are being managed as a collection of zipped files, it becomes necessary for the Access engine to make sure that the files are unzipped and available for the ReadOFS utility. Hence this becomes an added responsibility to the Access Engine layer which typically handles only the connections with the external engines. Since the data preparation is not handled by the existing Access Engines of HIDE, a specialized Access Engine – OFSAccessEngine is plugged into the Access Engine layer through the PluginAccessEngine (plugin for the AccessEngine) of the layer. The integration at the Access Engine layer can be represented as shown figure 7.6

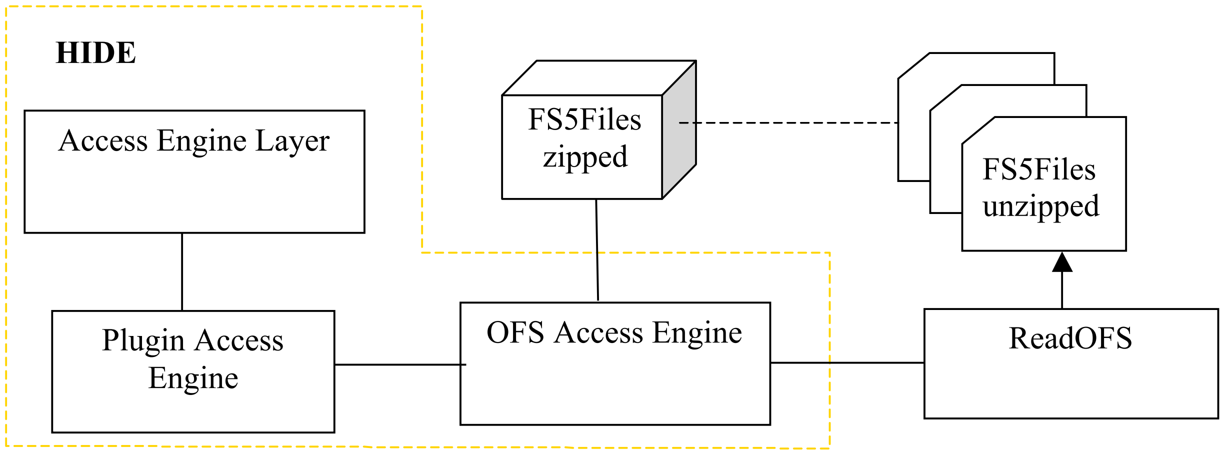


Figure 7.6 The integration of FS5Files to the Access Engine layer. The relations between various modules is also shown.

The responsibility of the Engine includes,

1. Data Preparation

   The searching and finding the appropriate zipped FS5File, and unzip it while making it available for the ReadOFS utility.
2. Prepare the request based on user entered values and parameters specified in the XAD files.
3. Handle the connection with ReadOFS utility.
4. Retrieve the results from ReadOFS which is external to HIDE.

7.3.2.Integration to Transformation Engine Layer

The time series data of FS5Files are temporal in nature. Hence it follows the temporal model of the HIDE system. The layer uses XFD metadata to recognize the syntactic structure of the retrieved data file (not the FS5File as it is not exposed to the HIDE system), and the XTMS metadata to understand the retrieved file as shown in Figure 7.5.

7.3.3.Integration to Search Engine and Query Engine

As mentioned in the preceding sections, the search engine utilizes the indices specified in the description model metadata files for the search of the appropriate Data node. Hence if the user has entered a string “Ohio RFC data” in the search engine, it will traverse the tree from “root” DataNode to the “OhioRFC” DataNode. Similarly, if the user entered string is “FS5Files data”, the Search engine will traverse from “root” node to the “FS5Files” DataNode. Various indices relating to different possibilities of search can be specified in the XDMS metadata model for each DataNode. i.e. FS5Files, NWSRFC, OhioRFC.

The Query Engine of the HIDE system utilizes the query model metadata file (XOMS) to identify the query plan to be used for querying the external engine (ReadOFS). This file is defined for the DataNode “Ohio RFC.” If a need occurs for aggregating the time series data from multiple RFC’s, a query model is to be defined for the DataNode “NWSRFC” as well. The information in the query model for the DataNode “Ohio RFC” includes,

1. List of search parameters:
    a. Time Series ID
    b. Data Type ID
    c. Number of periods to be retrieved
    d. Date and time
    e. Units of Data to be considered (English/Metrics)
2. Validation rule for parameters period and Date
    a. The range of periods is limited from 1 – 23
    b. The Date should follow a format of “YYYY MM DD HH”
3. The output parameter OFS time series data.

Query engine dynamically generates a query interface based on the query metadata model, validates the user entered values and send the request to the Access Engine.

Chapter 8 .CONCLUSIONS AND FUTURE WORK

8.1. Summary

Scientific data integration is a pressing issue in today's global scenario. Multitudes of data residing over various locations and being controlled by scientific institutions pose many challenges in the scientific data integration problem. The heterogeneity of data and sources with respect to the syntax, semantics, and structure imposes significant impact on the architecture and the design of any data integration development. As the data integration solutions move towards third generation of information systems, using semantics for data search and retrieval provides an easier search and access for the end users. The use of metadata as a context for the data is one of the many solutions used in the current information systems.

In this work, a novel model for integration of heterogeneous hydrologic data engines which are autonomous and diverse, using the concepts of data nodes and data node tree is presented. The heterogeneities of the sources are addressed through a tree model which incorporates semantics and structural nature of the data, while syntactical information of the data is handled through the metadata model. HIDE uses an extended XML-based wrapper/mediator, layered architecture for the integration, which realizes the data node tree, performs the query operations by accessing the external data system, and transforms them to the data models of the HIDE system. The problems of integration are systematically handled at each layer. An XML-based metadata is used for the definition of the DataNodes, operations, transformations and accessibility to the external data source/engine. HIDE uses temporal and spatial model for the representation of the data as opposed to the data models as structures, arrays or grids used in other geographic systems. This is based on the nature of the hydrology data in the time space domain. Finally HIDE encompasses the objectives of flexibility, scalability and simplicity which are fundamental to software system, in its design and implementation.

8.2. Future Work

While this thesis has produced a scalable, flexible and simple data integration model for the autonomous heterogeneous datasets, there is much scope for future research in this topic. Some potential areas of improvement are discussed below.

8.2.1. Metadata Security Mechanisms

Metadata is a crucial component in any data system. This work has demonstrated the significant role the metadata plays in a data integration scenario. Although metadata is just "data about data," it contains important information about the data. In fact, an unauthorized change to the metadata could be a serious attack on the data system itself. Hence securing the metadata becomes an important concern in the data integration.

In the HIDE system, metadata is used as a point of contact with external data sources. Hence sufficient measures are to be made to secure the metadata to prevent any insecure access to the external data sources. The need for metadata confidentiality, privacy, authentication and metadata integrity needs to be addressed in the system. However, providing a tight security for the data integration and loose security for the data sources does not necessarily solve the unauthorized release of information. In order to strike off a balance with sufficiency and necessity of the security measures, a study has been started, to use secret key encryption methodologies for the metadata and a scheme of partial user authentication with Merkle Trees. As the DataNode tree represents a logical hierarchy of information, it can be used to identify Merkle Trees. The use of secret keys to define the ownership for each DataNode by a source helps in user authentication using Merkle Trees. This is more advantageous, as it can determine, with less storage of intermediate states, if the DataNode tree has been altered. Current studies are continued in this particular area.

8.2.2. HIDE-gen : A metadata generation tool

Keeping in mind the objectives of simplicity and flexibility, this work has demonstrated the metadata model and effective usage of it in the data integration. However generating a metadata model and DataNode tree can be a "bottleneck" in the system, if they are not created properly. An Incorrect DataNode tree can result in incorrect representation of the information space and can consume time and effort in a search process. Also, incorrect metadata model will prohibit the user from retrieving the data from the external data sources correctly.

In order to address these issues, a metadata generation tool for the HIDE system is required. This tool can help the user to generate the metadata and in turn the DataNode tree based on predefined constraints. As the interaction of the user with the metadata can be limited to a certain extent with this tool, it might provide a secure mechanism for the creation of the metadata. A combination of the tool and secure mechanism can ensure the best mechanisms for the data integration.

8.2.3. Data Visualization

Data Visualization in 2D and 3D models is very much essential for an effective data analysis. At present, HIDE allows the user to view the data which are being retrieved from various data sources. The porting of a data visualization tool to HIDE will be done in the near future.

8.2.4. Data Mining Models

Data mining exercises on observational data sets is useful in extracting hidden/unknown relationships. For example in case of time series data, such analysis facilitates in deriving recurrent patterns. This process of locating efficient and accurate patterns of interest in time series data is important in data analysis in scientific domain. One such example would be to use this information for weather prediction analysis in hydrology domain.

The time series models are essential in the hydrology domain as it deals with non-stationary multivariate time series data on a daily basis. Traditional time series modelling is based on global linear models. One approach is Box-Jenkins family of models, where the current value is modeled as a weighted linear combination of past values plus additive noise and is often used in time series prediction. Thus incorporating data mining models with the data models of HIDE would be advantageous in analyzing forecasting nature of the time series data.

Appendix

XML Schema for the metadata model

Documentation:hide-funcModel

This metadata represents the description model of the HIDE system. It describes each data source and its child members along with a comprehensive description.

Details:

Recommended Usage	All data sources , data nodes and concepts
StandAlone	Yes
Schema name	funcModel.xsd

Element Definitions:

1. DataNode

Default Value	No
Type	A complex type with a sequence of other elements
Description	Describe the data node
Minimum Occurrence	1
Maximum Occurrence	1

Schema:

```
<xs:element name="DataNode">
        <xs:complexType>
                <xs:sequence>
                        <xs:element ref ="identifiedAs"/>
                        <xs:element ref="documentedBy "/>
                        <xs:element ref ="canMeasure"/>
                        <xs:element ref ="supportOperations"/>
                        <xs:element ref ="DataNodeMembers" />
                </xs:sequence>
        </xs:complexType>
</xs:element>
```

2. identifiedAs

Default Value	No
Optional	No
Type	A complex type with a collection of other elements such as name, label and index.
Description	Describe the identity of the data node. Name has to be unique as it is used for identifying the node. Index should be comma separated indices of keywords used for search.
Minimum Occurrence	1
Maximum Occurrence	1

Schema:

```
<xs:element  name="identity" minOccurs=”1” maxOccurs=”1”>
    <xs:all>
        <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="label" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="index" minOccurs="1" maxOccurs="1" type="xs:string"/>
    </xs:all>
</xs:element>
```

3. documentation

Default Value	No
Optional	Yes
Type	A complex type with a collection of other elements such as url and history. URL has to be a valid url to a description of the data source. The history should be a brief textual description about the source.

Description	Describe the documentation related to the data node
Minimum Occurrence	0
Maximum Occurrence	1

Schema:

```
<xs:element  name="documentation" minOccurs=”0” maxOccurs=”1”>
       <xs:all>
            <xs:element name="url" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="history" type="xs:string" minOccurs="1"
maxOccurs="1"/>
        </xs:all>
</xs:element>
```

4. canMeasure

Default Value	No
Optional	Yes
Type	Measurements.
Description	Describe the measurements collected at each DataNode.
Minimum Occurrence	0
Maximum Occurrence	1

Schema:

```
<xs:element name="measurements" minOccurs=”0” maxOccurs=”1”>
     <xs:all>
        <xs:element name="metadata" type="xs:string" minOccurs="1" maxOccurs="1"/>
       <xs:element name="measurements" minOccurs="1" maxOccurs="1">
            <xs:complexType>
                 <xs:sequence>
                      <xs:element ref ="measurement"/>
                </xs:sequence>
            </xs:complexType>
```

```
    </xs:element>
  </xs:all>
</xs:element>
```

5. measurement

Default Value	No
Optional	Yes
Type	A complex type with a sequence of elements such as type, minValue and maxValue. The type represents the data type for the measurement (int, float etc.). The minValue and maxValue are the thresholds of the measurements
Description	Describe the measurement type of DataNode.
Minimum Occurrence	0
Maximum Occurrence	Unbounded

Schema:

```
<xs:element name="measurement" minOccurs=”0” maxOccurs=”unbounded”>
    <xs:sequence>
        <xs:element name="constraint" minOccurs="0" maxOccurs="1">
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="type" type="xs:string" minOccurs="1"
maxOccurs="1"/>
                    <xs:element name="minValue" type="xs:float" minOccurs="1"
maxOccurs="1"/>
                    <xs:element name="maxValue" type="xs:float" minOccurs="1"
maxOccurs="1"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
    <xs:attribute name="name"/
</xs:element>
```

6. support Operations

Default Value	No
Optional	No
Type	A complex type with a sequence of elements defining the operations. The operations can be select ("query") and get. For the select operations, element points to the query operational metadata file.
Description	Describe the supported operations in a DataNode.
Minimum Occurrence	0
Maximum Occurrence	1

Schema:

```
<xs:element name="operations" minOccurs="0" maxOccurs="1">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="operation" minOccurs="1" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>

<xs:element name="operate">
    <xs:complexType>
      <xs:sequence>
            <xs:element name="select" minOccurs="0" maxOccurs="1"
type="selectType"/>
            <xs:element name="get" type="xs:string" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
    <xs:attribute name="type" type="xs:string"/>
  </xs:complexType>
</xs:element>

<xs:complexType name="selectType">
    <xs:all>
        <xs:element name="select_templates">
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="select_template" minOccurs="1"
maxOccurs="unbounded">
                        <xs:complexType>
                            <xs:attribute name="fileName" type="xs:string"/>
                            <xs:attribute name="seqNo" type="xs:integer" default="0"/>
                        </xs:complexType>
                      </xs:element>
                </xs:sequence>
              </xs:complexType>
           </xs:element>
        </xs:all>
        <xs:attribute name="templates" type="xs:string" default="yes"/>
</xs:complexType>
```

7. DataNodeMember

Default Value	No
Optional	Yes
Type	A complex type with a sequence of DataNodes.
Description	List the entire child DataNodes. Each DataNode element will point to the description model metadata of the DataNode.
Minimum Occurrence	0

Maximum Occurrence	1

Schema:

```
<xs:element name="DataNodeMembers" minOccurs="0" maxOccurs="1">
        <xs:complexType>
                <xs:sequence>
                <xs:element name="DataNodeMember" type="xs:anyURI"
minOccurs="1" maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
</xs:element>
```

Documentation: hide-querymodel

This metadata represents the query model of the HIDE system. It describes the query conditions, search parameters and validation rules to be applied to the condition.

Details:

Recommended Usage	Define for DataNodes which requires a querymodel. Mandatory for the leaf DataNode.
StandAlone	No. Dependent on the description model
Schema name	Query_template.xsd

Element Definitions:

1. Query

Default Value	No
Optional	Yes
Type	A complex type with a sequence of external or internal query. If the integration is of primary level, then query is an internal query. If the integration is of secondary level, the query is an

	external query.
Description	The query templates are described
Minimum Occurrence	1
Maximum Occurrence	1

Schema:

```
<xs:element name="Query">
    <xs:complexType>
        <xs:choice>
                <xs:element name="external" type="external"/>
                <xs:element name="internal" type="internal"/>
        </xs:choice>
    </xs:complexType>
</xs:element>
```

2. External query

Default Value	No
Optional	Yes , if internal query is used
Type	A complex type with URL and description. The URL points to the external interface of the data source. The description is a textual information about the data source.
Description	Definition of the external query
Minimum Occurrence	1
Maximum Occurrence	1

Schema:

```
<xs:complexType name="external">
    <xs:all>
        <xs:element name="details">
        <xs:complexType>
                <xs:all>
                        <xs:element name="URL" type="xs:anyURI"/>
```

```
                    <xs:element name="description" type="xs:string"/>
               </xs:all>
          </xs:complexType>
          </xs:element>
     </xs:all>
</xs:complexType>
```

3. internal query

Default Value	No
Optional	Yes, if external query is used.
Type	A complex type with search condition and output criteria elements
Description	Definition of the external query
Minimum Occurrence	1
Maximum Occurrence	1

Schema:

```
<xs:complexType name="internal">
     <xs:sequence>
          <xs:element ref="search_criteria" minOccurs="1" maxOccurs="1"/>

          <xs:element ref="output_criteria" minOccurs="0" maxOccurs="1"/>
     </xs:sequence>
</xs:complexType>
```

4. search criteria

Default Value	No
Optional	No
Type	A complex type with fields and validation rule elements
Description	Definition of the search criteria

Minimum Occurrence	1
Maximum Occurrence	1

Schema:

```
<xs:element name="search_criteria">
        <xs:complexType>
                <xs:sequence>
                        <xs:element ref="fields" minOccurs="1" maxOccurs="1"/>
                </xs:sequence>
        </xs:complexType>
</xs:element>
```

5. Fields

Default Value	No
Optional	No
Type	A Sequence of different types of field (Choice, Entry) with parameters
Description	Definition of the Fields
Minimum Occurrence	1
Maximum Occurrence	1

Schema:

```
<xs:element name="fields">
    <xs:sequence>
        <xs:element name="field" minOccurs="1" maxOccurs="unbounded">
            <xs:complexType>
                <xs:choice>
                        <xs:element ref="choice"/>
                        <xs:element ref="Entry"/>
                </xs:choice>
                <xs:attribute name="name" type="xs:string"/>
                <xs:attribute name="displayName" type="xs:string"/>
                <xs:attribute name="description" type="xs:string"/>
                <xs:attribute name="format" type="xs:string"/>
                <xs:attribute name="unit" type="xs:string"/>
                <xs:attribute name="type" type="fieldTypes"/>
                <xs:attribute name="empty" type="xs:string" default="yes"/>
                <xs:attribute name="dataNodeSearch" type="xs:string" default="no"/>
                <xs:attribute name="boundsCheck" type="xs:string" default="no"/>
                <xs:attribute name="relatedTo" type="xs:string" default="norelation"/>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
</xs:element>
```

6. choice element

Default Value	No
Optional	-
Type	A complex type with items.
Description	Definition of the choice field. Choice field can take inputs from a separate file or as a series values defined in the metadata.
Minimum Occurrence	-
Maximum Occurrence	-

Schema:

```
<xs:element name="choice">
    <xs:complexType>
        <xs:sequence>
                <xs:element name="item" minOccurs="0" maxOccurs="unbounded">
                        <xs:complexType>
                                <xs:attribute name="name" type="xs:string"/>
                                <xs:attribute name="value" type="xs:string"/>
                                <xs:attribute name="displayString" type="xs:string"/>
                        </xs:complexType>
                </xs:element>
        </xs:sequence>
        <xs:attribute name="type" type="choiceTypes"/>
        <xs:attribute name="input" type="inputTypes"/>
        <xs:attribute name="value" type="xs:anyURI"/>
        <xs:attribute name="defaultValue" type="xs:string" default="0"/>
    </xs:complexType>
</xs:element>
```

7. Entry

Default Value	No
Optional	-
Type	A complex type
Description	Definition of the Entry field. Entry field can be single entry field, Range field (min-max), Array field and spatial array entry field for defining spatial array as lat:long
Minimum Occurrence	-
Maximum Occurrence	-

Schema:

```
<xs:element name="Entry">
        <xs:complexType>
                <xs:choice>
                        <xs:element ref="SingleEntry"/>
                        <xs:element ref="Range"/>
                        <xs:element ref="ArrayEntry"/>
                        <xs:element ref="SpatialArrayEntry"/>
                </xs:choice>
                <xs:attribute name="type" type="entryFieldType"/>
                <xs:attribute name="defaultValue" type="xs:string" default="0"/>
        </xs:complexType>
```

```
</xs:element>
```

8. Integer data type field.

Default Value	No
Optional	-
Type	A complex type
Description	Definition of the integer datatype field representing an integer. This datatype field is defined for entry fields.
Minimum Occurrence	-
Maximum Occurrence	-

Schema:

```
<xs:element name="Integer">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="ValidationRule" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="rule" type="booleanTypes"/>
        <xs:attribute name="name" type="xs:string"/>
    </xs:complexType>
</xs:element>
```

9. String datatype field

Default Value	No
Optional	-
Type	A complex type
Description	Definition of the string datatype field representing a string. This datatype field is defined for entry fields.
Minimum Occurrence	-

Maximum Occurrence	-

Schema:

```
<xs:element name="String">
    <xs:complexType>
        <xs:sequence>
                <xs:element ref="ValidationRule" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="rule" type="booleanTypes"/>
    </xs:complexType>
</xs:element>
```

10. Date datatype field

Default Value	No
Optional	-
Type	A complex type
Description	Definition of the date datatype field representing a date. This datatype field is defined for entry fields (Single and range).
Minimum Occurrence	-
Maximum Occurrence	-

Schema:

```
<xs:element name="Date">
    <xs:complexType>
        <xs:sequence>
                <xs:element ref="ValidationRule" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="format" type="DateType"/>
        <xs:attribute name="rule" type="booleanTypes"/>
    </xs:complexType>
</xs:element>
```

11. Validation rule

Default Value	No
Optional	-
Type	A complex type
Description	Definition of the validation rules. Different rules are range rule, array rule, spatial rule, pattern rule, format rule, date rule.
Minimum Occurrence	0
Maximum Occurrence	1

Schema:

```
<xs:element name="ValidationRule">
    <xs:complexType>
        <xs:choice>
            <xs:element name="RangeRule">
                <xs:complexType>
                    <xs:all>
                        <xs:element name="from" type="xs:string"/>
                        <xs:element name="to" type="xs:string"/>
                    </xs:all>
                </xs:complexType>
            </xs:element>
            <xs:element name="ArrayRule">
                <xs:complexType>
                    <xs:all>
                        <xs:element name="from" type="xs:string"/>
                        <xs:element name="to" type="xs:string"/>
                    </xs:all>
                </xs:complexType>
            </xs:element>
            <xs:element name="SpatialRule">
                <xs:complexType>
```

```
                    <xs:all>
                        <xs:element name="from" type="xs:string"/>
                        <xs:element name="to" type="xs:string"/>
                    </xs:all>
                  </xs:complexType>
                </xs:element>
                <xs:element name="PatternRule" type="xs:string"/>
                <xs:element name="FormatRule" type="xs:string"/>
                <xs:element name="DateRule">
                   <xs:complexType>
                     <xs:attribute name="format" type="xs:string"/>
                     <xs:attribute name="from" type="xs:string"/>
                     <xs:attribute name="to" type="xs:string"/>
                  </xs:complexType>
                </xs:element>
                </xs:choice>
                <xs:attribute name="type" type="ruleTypes"/>
         </xs:complexType>
</xs:element>
```

12. Output_criteria

Default Value	No
Optional	-
Type	A complex type
Description	Definition of the output criteria. Sometime, output criteria might depend on search conditions. This dependency can be represented as the attribute “dependsOn.”
Minimum Occurrence	0
Maximum Occurrence	1

Schema:

```
<xs:element name="output_criteria" minOccurs="0" maxOccurs="1">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="items">
                <xs:complexType>
                    <xs:sequence>
                       <xs:element ref="item" minOccurs="1"
maxOccurs="unbounded"/>
                    </xs:sequence>
                    <xs:attribute name="order" type="booleanTypes"/>
                    <xs:attribute name="dependsOn" type="xs:string" default="null"/>
                    <xs:attribute name="fileName" type="xs:string" default="null"/>
                </xs:complexType>
             </xs:element>
        </xs:sequence>
        <xs:attribute name="name" type="xs:string"/>
        <xs:attribute name="displayName" type ="xs:string"/>
   </xs:complexType>
</xs:element>
```

Documentation: hide-syntax

This metadata represents the syntax definitions of data in HIDE system. It describes the various data representations as ASCII, binary etc.

Details:

Recommended Usage	Define for DataNodes which has a querymodel.
StandAlone	No. Dependent on the query model
Schema name	Query_template.xsd

Element Definitions:

1. Syntax

Default Value	No
Optional	No
Type	A complex type
Description	Definition of the syntax. Currently, only ASCII is defined here. Syntax consist of comment fields, header fields and dataset fields
Minimum Occurrence	1
Maximum Occurrence	1

Schema:

```
<xs:element name="syntax">
    <xs:complexType>
        <xs:choice>
            <xs:element name="Ascii">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="Comment" minOccurs="0" maxOccurs="1">
                            <xs:complexType>
                                <xs:attribute name="value" type="xs:string"/>
                            </xs:complexType>
                        </xs:element>
                        <xs:element name="Header" minOccurs="1" maxOccurs="1">
                            <xs:complexType>
                                <xs:attribute name="type" type="xs:string"/>
                                <xs:attribute name="numLines" type="xs:string"/>
                            </xs:complexType>
                         </xs:element>
                         <xs:element ref="Dataset"/>
                        </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:choice>
        <xs:attribute name="type" type="xs:string"/>
        <xs:attribute name="delimiter" type="xs:string"/>
    </xs:complexType>
</xs:element>
```

2. Dataset

Default Value	No
Optional	No
Type	A complex type
Description	Definition of the dataset. Dataset can be a dataTable or dataArray
Minimum Occurrence	1
Maximum Occurrence	1

Schema:

```
<xs:element name="Dataset">
    <xs:complexType>
        <xs:choice>
            <xs:element name="dataTable">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element ref="dataFields" minOccurs="1" maxOccurs="1"/>
                    </xs:sequence>
                    <xs:attribute name="numRecords" type="xs:string"/>
                </xs:complexType>
            </xs:element>
            <xs:element name="dataArray">
                 <xs:complexType>
                     <xs:sequence>
                        <xs:element ref="dataFields" minOccurs="1" maxOccurs="1"/>
                    <xs:element ref="ArrDimensions" minOccurs="1"
maxOccurs="1"/>
                  </xs:sequence>
                  <xs:attribute name="numDimensions" type="xs:integer"/>
                </xs:complexType>
              </xs:element>
            </xs:choice>
            <xs:attribute name="type" type="xs:string"/>
          </xs:complexType>
</xs:element>
```

3. DataFields

Default Value	No
Optional	No
Type	A complex type
Description	Definition of the dataFields. DataFields represents each field in the data. Each data field has name, display name, its associated query field name, description and datatype associated with it.
Minimum Occurrence	1
Maximum Occurrence	1

Schema:

```
<xs:element name="dataFields">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="dataField" minOccurs="1" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:attribute name="name" type="xs:string"/>
                    <xs:attribute name="displayName" type="xs:string"/>
                    <xs:attribute name="type" type="xs:string"/>
                    <xs:attribute name="description" type="xs:string"/>
                    <xs:attribute name="query_fldname" type="xs:string"
use="optional"/>
                    <xs:attribute name="default" type="xs:string" default="no"/>
                  </xs:complexType>
               </xs:element>
            </xs:sequence>
     </xs:complexType>
</xs:element>
```

4. Array Dimensions

Default Value	No
Optional	No
Type	A complex type
Description	Definition of the Array Dimensions. This element defines the array dimensions for the data Array field
Minimum Occurrence	1
Maximum Occurrence	1

Schema:

```
<xs:element name="ArrDimensions">
     <xs:complexType>
         <xs:sequence>
                  <xs:element name="ArrDimension" minOccurs="1" maxOccurs="3">
                       <xs:complexType>
                           <xs:attribute name="name" type="xs:string"/>
                           <xs:attribute name="query_fldname" type="xs:string"/>
                       </xs:complexType>
                   </xs:element>
         </xs:sequence>
         <xs:attribute name="numDimensions" type="xs:string"/>
```

```
    </xs:complexType>
</xs:element>
```

Documentation: hide-Access Definition

This metadata represents the Access definitions for the HIDE system. The access specifications currently are defined for http and plugins.

Details:

Recommended Usage	Define for DataNodes which has a querymodel.
StandAlone	No. Dependent on the query model
Schema name	Query_template.xsd

Element Definitions:

1. AP Definition

Default Value	No
Optional	No
Type	A complex type
Description	Definition of Access point.
Minimum Occurrence	0
Maximum Occurrence	1

Schema:

```
<xs:element name="APDefinition" minOccurs="0" maxOccurs="1">
        <xs:complexType>
                <xs:choice>
                        <xs:element ref="http"/>
                        <xs:element ref="external_plugin"/>
                </xs:choice>
                <xs:attribute name="type" type="xs:string"/>
        </xs:complexType>
</xs:element>
```

2. HttpAccessEngine

Default Value	No
Optional	Yes, if plugin access engine is used.
Type	A complex type
Description	Definition of http Access point. It is a combination of the base URL with a set of selection and projection parameters.
Minimum Occurrence	0
Maximum Occurrence	1

Schema:

```
<xs:element name="http">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="baseURL" type="xs:anyURI" minOccurs="1"
maxOccurs="1"/>
                <xs:element ref="selection" />
        </xs:sequence>
      </xs:complexType>
</xs:element>
```

3. Selection

Default Value	No
Optional	No
Type	A complex type
Description	Definition of selection element. It consists of a set of selection parameters with name, value and corresponding query field name.

Minimum Occurrence	0
Maximum Occurrence	1

Schema:

```
<xs:element name="selection" minOccurs="0" maxOccurs="1">
    <xs:all>
        <xs:element ref="parameters"/>
            <xs:complexType>
                <xs:sequence>
                    <xs:element ref="parameter" />
                </xs:sequence>
                <xs:attribute name="empty" type="booleanTypes" default="yes"/>
            </xs:complexType>
        </xs:element>
    </xs:all>
</xs:element>
```

4. Parameter

Default Value	No
Optional	No
Type	A complex type
Description	Definition of parameter element.
Minimum Occurrence	1
Maximum Occurrence	unbounded

Schema:

```
<xs:element name="parameter" minOccurs="1" maxOccurs="unbounded">
        <xs:complexType>
                <xs:attribute name="type" type="xs:string" default="default"/>
                <xs:attribute name="param" type="xs:string"/>
                <xs:attribute name="value" type="xs:string"/>
                <xs:attribute name="defaultValue" type="xs:string"/>
        </xs:complexType>
```

```
</xs:element>
```

5. Plugin AccessEngine

Default Value	No
Optional	No
Type	A complex type
Description	Definition of plugin Access Engine.
Minimum Occurrence	1
Maximum Occurrence	Unbounded

Schema:

```
<xs:element name="external_plugin">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="plugin"/>
                <xs:complexType>
                    <xs:attribute name="name" type="xs:string"/>
                     <xs:attribute name="directory" type="xs:anyURI"/>
                </xs:complexType>
             </xs:element>
            <xs:element name="selection" type="selection" minOccurs="0"
maxOccurs="1"/>
        </xs:sequence>
     </xs:complexType>
</xs:element>
```

Documentation: hide-Transformation

This metadata represents the DataModel transformation definitions for the HIDE system. The transformation definitions includes the type of DataModel to be considered and the temporal and geographic coverage

Details:

Recommended Usage	Define for DataNodes which has a querymodel.
StandAlone	No. Dependent on the query model
Schema name	TESchema.xsd

Element Definitions:

1. TE

Default Value	No
Optional	No
Type	A complex type
Description	Definition of transformation engine. The engine can be spatial model or temporal model
Minimum Occurrence	1
Maximum Occurrence	1

Schema:

```
<xs:element name="TE">
    <xs:complexType>
        <xs:all>
            <xs:element name="TEDataModelType">
                <xs:complexType>
                    <xs:choice>
                        <xs:element ref="TimeModelTE"/>
                        <xs:element ref="SpatialModelTE"/>
                    </xs:choice>
                    <xs:attribute name="type" type="xs:string"/>
                </xs:complexType>
            </xs:element>
        </xs:all>
    </xs:complexType>
```

</xs:element>

2. TimeModelTE

Default Value	No
Optional	Yes, if spatial model is defined
Type	A complex type
Description	Definition of time model transformation engine. The data would be represented as a dataTable
Minimum Occurrence	1
Maximum Occurrence	1

Schema:

```
<xs:element name="TimeModelTE">
        <xs:complexType>
            <xs:sequence>
                <xs:choice>
                     <xs:element ref="dataTable"/>
                </xs:choice>
                <xs:element ref="SpatialCoverage" minOccurs="0" maxOccurs="1"/>
                <xs:element ref="TemporalCoverage" minOccurs="0"
maxOccurs="1"/>
            </xs:sequence>
            <xs:attribute name="datasetType" type="xs:string"/>
        </xs:complexType>
</xs:element>
```

3. SpatialModel TE

Default Value	No
Optional	Yes, if temporal model is defined
Type	A complex type
Description	Definition of spatial model transformation engine. The data would be represented as a dataTable or dataArray

Minimum Occurrence	1
Maximum Occurrence	1

Schema:

```
<xs:element name="SpatialModelTE">
    <xs:complexType>
        <xs:sequence>
            <xs:choice>
                <xs:element ref="dataTable"/>
                <xs:element ref="dataArray"/>
            </xs:choice>
            <xs:element ref="SpatialCoverage" minOccurs="0" maxOccurs="1"/>
            <xs:element ref="TemporalCoverage" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="datasetType" type="xs:string"/>
    </xs:complexType>
</xs:element>
```

4. Temporal Coverage

Default Value	No
Optional	Yes
Type	A complex type
Description	Definition of temporal coverage as a range of date.
Minimum Occurrence	0
Maximum Occurrence	1

Schema:

```
<xs:element name="TemporalCoverage">
        <xs:complexType>
                <xs:all>
                        <xs:element name="Date">
                            <xs:complexType>
```

```
                              <xs:attribute name="start" type="xs:string"/>
                              <xs:attribute name="end" type="xs:string"/>
                         </xs:complexType>
                    </xs:element>
               </xs:all>
          </xs:complexType>
</xs:element>
```

5. Spatial Coverage

Default Value	No
Optional	Yes
Type	A complex type
Description	Definition of spatial coverage as a bounding rectangle with N,S,E,W latitudes and longitudes
Minimum Occurrence	0
Maximum Occurrence	1

Schema:

```
<xs:element name="SpatialCoverage">
     <xs:complexType>
          <xs:all>
               <xs:element name="Latitude">
                    <xs:complexType>
                         <xs:attribute name="North" type="xs:integer"/>
                         <xs:attribute name="South" type="xs:integer"/>
                    </xs:complexType>
               </xs:element>
               <xs:element name="Longitude">
                    <xs:complexType>
                         <xs:attribute name="East" type="xs:integer"/>
                         <xs:attribute name="West" type="xs:integer"/>
                    </xs:complexType>
```

```
        </xs:element>
      </xs:all>
      <xs:attribute name="resolution" type="xs:string" default="1"/>
   </xs:complexType>
</xs:element>
```

6. dataArray

Default Value	No
Optional	Yes, if dataTable is used
Type	A complex type
Description	Definition of dataArray
Minimum Occurrence	0
Maximum Occurrence	1

Schema:

```
<xs:element name="dataArray">
      <xs:complexType>
            <xs:attribute name="dimension" type="xs:string"/>
      </xs:complexType>
</xs:element>
```

7. dataTable

Default Value	No
Optional	Yes, if dataArray is used
Type	A complex type
Description	Definition of dataTable
Minimum Occurrence	0
Maximum Occurrence	1

Schema:

```
<xs:element name="dataTable">
      <xs:complexType>
            <xs:all>
```

```
                    <xs:element name="TimeFormat" type="xs:string"/>
                </xs:all>
        </xs:complexType>
</xs:element>
```

Documentation: hide-dataModel

This schema defines the DataModels of the HIDE system. The information includes the mathematical model, representational model and the information model.

Details:

Recommended Usage	Define for the datasets retrieved.
StandAlone	Yes
Schema name	dataObjectModel_schema.xsd

Element Definitions:

1. DataModel

Default Value	No
Optional	No
Type	A complex type
Description	Definition of DataModel. The model contains information model, representation model and mathematical model. Based on the mathematical model, the data model can be temporal / spatial.
Minimum Occurrence	1
Maximum Occurrence	1

Schema:

```
<xs:element name="dataModelObject">
        <xs:complexType>
                <xs:sequence>
                        <xs:element ref="DataSources"/>
                        <xs:element ref="DataSet"/>
                </xs:sequence>
                <xs:attribute name="type" type="xs:string"/>
          </xs:complexType>
</xs:element>
```

2.DataSources

Default Value	No
Optional	No
Type	A complex type
Description	Definition of DataSources.
Minimum Occurrence	1
Maximum Occurrence	1

Schema:

```
<xs:element name="DataSources">
   <xs:complexType>
     <xs:sequence>
         <xs:element name="DataSource" minOccurs="1" maxOccurs="unbounded">
             <xs:complexType>
                 <xs:sequence>
                         <xs:element name="name" type="xs:string"/>
                         <xs:element name="link" type="xs:anyURI"/>
                         <xs:element name="AdditionalInfo" type="xs:string"/>
                 </xs:sequence>
             </xs:complexType>
         </xs:element>
       </xs:sequence>
    </xs:complexType>
</xs:element>
```

3. Dataset

Default Value	No
Optional	No
Type	A complex type
Description	Definition of datasets. It consists of information about the coverage, representational information and the mathematical DataModels.
Minimum Occurrence	1
Maximum Occurrence	1

Schema:

```
<xs:element name="DataSet">
        <xs:complexType>
                <xs:sequence>
                        <xs:element ref="Syntax" minOccurs="1" maxOccurs="1"/>
                        <xs:element ref="Coverage" minOccurs="1" maxOccurs="1"/>
                        <xs:choice>
                                <xs:element ref="TemporalModel"/>
                                <xs:element ref="SpatialModel"/>
                                <xs:element ref="TemporalSpatialModel"/>
                        </xs:choice>
                </xs:sequence>
        </xs:complexType>
</xs:element>
```

4.Coverage

Default Value	No
Optional	No
Type	A complex type
Description	Definition of coverage. It consists of information about the temporal and geographical coverage of the data

Minimum Occurrence	1
Maximum Occurrence	1

Schema:

```
<xs:element name="Coverage" minOccurs="0" maxOccurs="1">
        <xs:complexType>
                <xs:sequence>
                        <xs:element ref="temporal" minOccurs="0" maxOccurs="1"/>
                        <xs:element ref="spatial" minOccurs="0" maxOccurs="1"/>
                </xs:sequence>
        </xs:complexType>
</xs:element>
```

5. Temporal Coverage

Default Value	No
Optional	Yes
Type	A complex type
Description	Definition of temporal coverage. The coverage can be provided as a range of dates with formats and time zones.
Minimum Occurrence	1
Maximum Occurrence	1

Schema:

```
<xs:element name="temporal">
   <xs:complexType>
      <xs:sequence>
        <xs:element name="from" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="to" type="xs:string" minOccurs="1" maxOccurs="1"/>
      </xs:sequence>
      <xs:attribute name="format" type="xs:string"/>
      <xs:attribute name="type" type="xs:string"/>
    </xs:complexType>
```

```
</xs:element>
```

6. Spatial Coverage

Default Value	No
Optional	Yes
Type	A complex type
Description	Definition of spatial coverage as a rectangle with bounding coordinates representing the latitudes and longitudes.
Minimum Occurrence	1
Maximum Occurrence	1

Schema:

```
<xs:element name="spatial">
   <xs:complexType>
       <xs:sequence>
          <xs:element name="northBoundCoordinate" type="xs:float" minOccurs="1"
maxOccurs="1"/>
          <xs:element name="southBoundCoordinate" type="xs:float" minOccurs="1"
maxOccurs="1"/>
          <xs:element name="eastBoundCoordinate" type="xs:float" minOccurs="1"
maxOccurs="1"/>
          <xs:element name="westBoundCoordinate" type="xs:float" minOccurs="1"
maxOccurs="1"/>
       </xs:sequence>
     </xs:complexType>
</xs:element>
```

7. Temporal Model

Default Value	No
Optional	Yes
Type	A complex type

Description	Definition of Temporal model. It consists of the start time, format of the time and the timesteps.
Minimum Occurrence	1
Maximum Occurrence	1

Schema:

```
<xs:element name="TemporalModel">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="dataFields" minOccurs="1" maxOccurs="1"/>
             <xs:element name="dateTime" minOccurs="1" maxOccurs="1">
                 <xs:complexType>
                   <xs:all>
                       <xs:element name="startValue"/>
                             <xs:complexType>
                                 <xs:attribute name="format" value="xs:string"/>
                             </xs:complexType>
                        </xs:element>
                        <xs:element ref="timeStep"/>
                   </xs:all>
                 </xs:complexType>
               </xs:element>
          </xs:sequence>
     </xs:complexType>
</xs:element>
```

8. TimeStep

Default Value	No
Optional	No
Type	A complex type
Description	Definition of Timesteps.
Minimum Occurrence	1

Maximum Occurrence	1

Schema:

```
<xs:element name="timeStep" minOccurs="1" maxOccurs="1">
        <xs:complexType>
                        <xs:attribute name="value" type="xs:integer"/>
        </xs:complexType>
</xs:element>
```

9. dataFields

Default Value	No
Optional	No
Type	A complex type
Description	Definition of the dataFields. The dataFields can be stationary or non-stationary.
Minimum Occurrence	1
Maximum Occurrence	1

Schema:

```
<xs:element name="dataFields">
    <xs:complexType>
        <xs:sequence>
                <xs:element name="stationary">
                        <xs:complexType>
                                <xs:sequence>
                                        <xs:element ref="dataField"/>
                                </xs:sequence>
                        </xs:complexType>
                </xs:element>
                <xs:element name="non-stationary">
                        <xs:complexType>
                                <xs:sequence>
                                        <xs:element ref="dataField"/>
                                </xs:sequence>
```

```
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
<xs:element>
```

10. dataField

Default Value	No
Optional	No
Type	A complex type
Description	Definition of the attributes
Minimum Occurrence	1
Maximum Occurrence	unbounded

Schema:

```
<xs:element name="dataField">
    <xs:complexType>
        <xs:attribute name="id" type="xs:string"/>
        <xs:attribute name="name" type="xs:string"/>
        <xs:attribute name="displayName" type="xs:string"/>
        <xs:attribute name="description" type="xs:string"/>
        <xs:attribute name="type" type="xs:string"/>
    </xs:complexType>
</xs:element>
```

11.SpatialModel

Default Value	No
Optional	No
Type	A complex type
Description	Definition of the spatial model. It contains the latitude and longitude range, resolution of the cell in the grid, dataFields.

Minimum Occurrence	1
Maximum Occurrence	1

Schema:

```
<xs:element name="SpatialModel">
        <xs:complexType>
                <xs:all>
                    <xs:element name="LatitudeRange">
                         <xs:complexType>
                                 <xs:all>
                                         <xs:element name="from"> type="xs:integer"/>
                                         <xs:element name="to"> type="xs:integer"/>
                                 </xs:all>
                         </xs:complexType>
                    </xs:element>
                    <xs:element name="LongitudeRange">
                         <xs:complexType>
                                 <xs:all>
                                         <xs:element name="from"> type="xs:integer"/>
                                         <xs:element name="to"> type="xs:integer"/>
                                 </xs:all>
                         </xs:complexType>
                   </xs:element>
                   <xs:element name="resolution">
                                 <xs:complexType>
                                         <xs:attribute name="unit" type="xs:string"/>
                                         <xs:attribute value="type" type="xs:float"/>
                                 </xs:complexType>
                </xs:element>
                <xs:element ref="dataFields"/>
        </xs:all>
</xs:complexType>
```

```
</xs:element>
```

12. Syntax

Default Value	No
Optional	No
Type	A complex type
Description	Definition of the representational model. It contains the information about how the data is stored in HIDE.
Minimum Occurrence	1
Maximum Occurrence	1

Schema:

```
<xs:element name="Syntax">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="Ascii">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="dataFields" minOccurs="1" maxOccurs="1"/>
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element name="id">
                                        <xs:complexType>
                                            <xs:attribute name="value" type="xs:string"/>
                                        </xs:complexType>
                                    </xs:element>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:complexType>
```

```
                    </xs:element>
                </xs:sequence>
                <xs:attribute name="type" type="xs:string"/>
            </xs:complexType>
        </xs:element>
```

REFERENCES

Anand, S.S., and Buchner, A.G. (1998). Decision support using data mining, Financial Times Pitman Publishers.

Arens, Y., Knoblock. C. (1993). SIMS: Retrieving and integrating information from multiple sources. ACM SIGMOD Record, 22(2), 562-563.

Arens, Y., Knoblock, C. A. and Shen, W.-M. (1996). Query reformulation for dynamic information integration. Journal of Intelligent Information Systems - Special Issue on Intelligent Information Integration, 6(2/3), 99-130.

Australian Antarctic Automatic Weather Station Dataset. (2004), http://www.antcrc.utas.edu.au/argos/awswebsite/datapage.html

Baru, C., Gupta,A., Ludäscher, B., Marciano, R., Papakonstantinou,Y., Velikhov,P., Chu, V.(1999). XML-based information mediation with MIX. SIGMOD Record, 28(2),597-599

Bayardo, R., Bohrer, W., Brice, R., Cichocki, A., Fowler, G., Helal, A., Kashyap, V., Ksiezyk, T., Martin, G., Nodine, M., Rashid, M., Rusinkiewicz, M., Shea, R., Unnikrishnan, C., Unruh, A., Woelk, D. (1997). InfoSleuth: agent-based semantic integration of information in open and dynamic environments. Proceedings of the ACM SIGMOD International Conference on Management of Data. (pp. 195-206).

Boucelma, O., Castano, S., Goble, C., Josifovski, V., Lacroix, Z., Ludäscher, B.(2002). Report on the EDBT'02 panel on scientific data integration. ACM SIGMOD Record, 31(4), 107-112.

Bowers, S., and Lud ascher, B.(2003). Towards a generic framework for semantic registration of scientific data. Semantic Web Technologies for Searching and Retrieving Scientific Data (SCISW).

Bergamaschi, S., Castano, S., Vincini, M. (1999). Semantic integration of semistructured and structured data sources. ACM SIGMOD Record, 28(1),54-59

Bray, T., Paoli, J., Sperberg-McQueen, C. M. (1998). The Extensible Markup Language (XML) 1.0. W3C recommendation, World Wide Web Consortium,http://www.w3.org/TR/1998/REC-xml-19980210

DODS (2004), Distributed Oceanographic system., *http://www.unidata.ucar.edu/packages/dods/*

ESML (2004), Earth Science Markup Language, *http://esml.itsc.uah.edu*

Egenhofer M.J.(2002). Toward the semantic geospatial web. Proceedings of the tenth ACM international symposium on Advances in geographic information systems, (pp.1-4)

EML (2004), Ecological Metadata Language, http://knb.ecoinformatics.org/software/eml

FGDC (2003), Federal Geographic Data Committee, http://www.fgdc.gov/metadata/metadata.html

Fonseca,F.T., Egenhofer, M.J (1999). Ontology-Driven Geographic Information Systems. Proceedings of the seventh ACM international symposium on Advances in geographic information systems.(pp. 14-19)

GML (2004), Geography Markup Language, http://opengis.net/gml

GPCC (2001), Global Climatology center, http://daac.gsfc.nasa.gov/CAMPAIGN_DOCS/FTP_SITE/INT_DIS/readmes/gpcp_global_precip.html

Gruber, Thomas (1993). Toward principles for the design of ontologies used for knowledge sharing**.** International Journal of Human-Computer Studies, Volume 43 (5), (pp. 907-928) special issue on the role of formal ontology in the information technology.

Gupta, A., Lud ascher, B., and Martone, M. E. (2002). Registering scientific information sources for semantic mediation. Proceedings of the 21st International Conference on Conceptual Modeling (ER), number 2503 in Lecture Notes in Computer Science (pp.182—198).

Hammer,J., García-Molina,H., Ireland, K., Papakonstantinou,Y., Ullman,J.,Widom. J. (1995). Information translation, mediation, and mosaic-based browsing in the TSIMMIS system. Proceedings of the 1995 ACM SIGMOD international conference on Management of data, 483

Hammer,J., García-Molina, H., Nestorov, S., Yerneni, R., Breunig,M., Vassalos,V.(1997). Template-based wrappers in the TSIMMIS system. Proceedings of the 1997 ACM SIGMOD international conference on Management of data, (pp. 532-535)

Jeong, S., Liang, Y. and Liang, X. (2004). Design of an integrated data retrieval, analysis, and visualization system: application in the hydrology domain, Environ. Modelling and Software, submitted.

JOSS (2004), Joint Office for Science Support, http://www.joss.ucar.edu

Levy,A., Srivastava, D. and Kirk. T.(1995). Data Model and Query Evaluation in Global Information Systems. Journal of Intelligent Information Systems (Vol. 5(2))

Levy,A., Rajaraman,A., and Ordille,J.J (1996). Querying heterogeneous information sources using source descriptions. Proceedings of International Conference on Very Large Databases (VLDB) , Bombay, India (pp. 251—262)

Levy, A., Mendelzon, A., Sagiv, Y., Srivastava, D. (1995). Answering queries using views. Proc. ACM PODS Symp.(pp.95—104)

Lin, K., and Ludäscher, B. (2003). A System for Semantic Integration of Geologic Maps via Ontologies, Semantic Web Technologies for Searching and Retrieving Scientific Data (SCISW), Sanibel Island, Florida.

Liu, Z., Liang, Y. and Liang, X. (2003). Integrated Data Management, Retrieval and Visualization System for Earth Science Datasets. Proceedings of the 17th Conference on Hydrology, Long Beach, California.

Mena, E., Illarramendi, A., Kashyap, V., and Sheth, P.(2000). OBSERVER: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. Distributed and Parallel Databases, (Vol. 8(2),pp. 223-271)

OGC (2004), Open Geospatial Consortium, http://www.opengeospatial.org

Pissinou, A, Makki, K, Park, E. K.(1993). Towards the design and development of a new architecture for Geographic Information Systems. Proceedings of the second international conference on Information and knowledge management, (pp. 565 – 573)

SEEK (2004), Science Environment for Ecological Knowledge, http://seek.ecoinformatics.org

SensorML (2004), Sensor Model Language. *http://vast.nsstc.uah.edu/SensorML/*

Sheth A 1998. Changing Focus on Interoperability in Information Systems: From System, Syntax, Structure to Semantics. In Interoperating Geographic Information Systems, M.F.GoodChild, M.J. Egenhofer, R.Fegeas, C.A.Kottman (Eds) Kluwer Publishers.

Shklar ,L.A., Sheth ,A.P., Kashyap ,V., Shah,K. (1995). InfoHarness: Use of Automatically Generated Metadata for Search and Retrieval of Heterogeneous Information. Proceedings of the 7th International Conference on Advanced Information Systems Engineering, (pp.217-230).

Shklar, L., Thatte, S., Marcus, H. and Sheth, A. The "Info Harness" Information Integration Platform. http://www. ncsa.uiuc.edu/SDG/ IT94/Proceedings/shklar/shklar.html.

Tomasic, A., Raschid, L., Valduriez, P.(1998). Scaling access to heterogeneous data sources with DISCO. IEEE Transactions on Knowledge and Data Engineering (pp. 808 – 823)

UCAR (2004),University Corporation for Atmospheric Research, http://www.ucar.edu/

Uitermark,H., Oosterom,P.V., Mars,N. and Molenaar,M (1999). Ontology-based Geographic Data Set Integration. Proceedings of STDBM'99, workshop on Spatio-Temporal Database Management, Edinburgh, Scotland (pp.60-79).

Unidata Program Center (UPC) (1998), NetCDF. http://www.unidata.ucar.edu/packages/netcdf/.

UN-SWI (1997), Comprehensive assessment of the freshwater resources of the world, United Nations and the Stockholm Water Institute, World Meteorological Organization, 33pp.

USGS (2004), U.S Geological Survey, *http://waterdata.usgs.gov/nwis*

Wiederhold, G. (1992). Mediators in the architecture of future information systems. IEEE Computer, 25(3),38—49

Zaslavsky, I., Marciano, R., Gupta, A., Baru, C. (2000).XML-based Spatial Data Mediation Infrastructure for Global Interoperability. 4th Global Spatial Data Infrastructure Conference, Cape Town, South Africa (pp.13-15).

Vita

Nimmy Ravindran was born in Cochin, India on May, 1975. She graduated with a Bachelor of Technology degree in Electronics Engineering from Cochin University, India in 1996. She pursued a career in Software development in the Computer Industry for 6 years. She worked as a Project Leader for Wipro Technologies, India, where she received a technical excellence award for her expertise in the performance tuning for the GSM-OMC system of Lucent Technologies. She then started her Masters in Electrical and Computer Engineering at Virginia Tech after taking a break from her career in Computer industry. She actively participated in the research activities of hydrologic information systems for National Weather Service. Her area of interest includes distributed data computing, Data Mining, and Data fusion.