Efficient Storage and Management of Environmental Information

Nabil R. Adam, Vijayalakshmi Atluri, and Songmei Yu
MSIS Department and CIMIC, Rutgers University
Newark, New Jersey 07102
{adam, atluri, songmei}@cimic.rutgers.edu

Yelena Yesha
Department of Computer Science and Electrical Engineering, UMBC
Baltimore, MD 21250
yeyesha@cs.umbc.edu

Abstract
Spatial Data warehouses pose many challenging requirements with respect to the design of the data model due to the nature of analytical operations and the nature of the views to be maintained by the spatial warehouse. The first challenge is due to the multi-dimensional nature of each dimension itself. In a traditional data warehouse the various dimensions contributing to the warehouse data are simple in nature, each having different attributes. Data models such as the star schema, fact constellation schema, snowflake schema or the multi-dimensional model, can therefore, be used to represent the traditional data warehouse. On the other hand, the different dimensions in a spatial data warehouse comprise of different types of data, each of which is multi-dimensional in nature. The current available data models are not adequate for such domains. In this paper, we propose a data model that is well suited for such domains, called the **cascaded star model** that is capable of representing multiple dimensions of a spatial data warehouse, where each dimension is multi-dimensional. The nature of the queries in such domains is different from that of traditional data warehouses (such as fly-by of a region), and therefore we propose a suitable architecture that allows specification of the queries and their visual presentation.

1. Introduction

In the area of Environmental and Earth sciences, we are concerned with collection, assimilation, cataloging and dissemination or retrieval of a vast array of environmental data. Environmental and Earth science computer systems receive their input from various types of satellite images with different resolutions captured by different sensors, models of the topography and spatial attributes of the landscape such as roads, rivers, parcels, schools, zip code areas, city streets and administrative boundaries (all exist in topographic maps), census information that describes the socio-economic and health characteristics of the population, processed digital terrain models into a new information product in the form of three-dimensional visualizations of digital terrain models projected as video ``fly-bys", and finally information transmitted (almost in real-time) from ground monitoring stations.

The system needs to provide flexible image extraction functionalities, such as hyper-spectral channel extraction, overlaying, and ad-hoc thematic coloring [4]. Such systems are intended to serve the evaluation and formulation of environmental policies by enabling users, including management and researchers to query various critical parameters such as ambient air and water quality and visualize the results in a graphical form. In addition to serving decision makers and researchers, these systems are intended to also serve the citizens, thus, enabling any citizen of any given district or a state to look at his/her county, community, home and be able to obtain relevant information on such issues as environment, health, and infrastructure, among others. Such systems should facilitate effective knowledge discovery in a manner tailored to changing needs and abilities of users, both intellectual and technological.

Consider for example the NASA Regional Application Center (RAC) at Rutgers Center for Information Management, Integration and Connectivity (CIMIC), which is a joint project between Rutgers CIMIC, NASA Goddard Space Flight Center (GSFC) and the New Jersey Meadowlands Commission (NJMC). As a RAC, CIMIC maintains a large collection of satellite images acquired through various sources. Specifically, the CIMIC-RAC currently stores and manages satellite imagery from various sources, including:

- Direct downloads of AVHRR data from polar orbiting satellites, such as NOAA 12, NOAA 14 and NOAA 15, over the Northeast region of the US including New York and New Jersey;
- LANDSAT and RADAR data obtained from NASA archives;
- Hyper-spectral images from the Airborne Imaging Spectrometer for Applications (AISA) sensor;
- Value-added products, such as AVHRR NDVI biweekly composites from the NASA EROS data center; Aerial ortho-photographs provided by various private companies; and
- Value-added products generated by various experts.

In addition to the images from a variety of space borne satellites, other data includes ground data from continuous monitoring weather stations, and maps, reports, data sets from federal, state and local government agencies. The problem is how to efficiently manage and store this diverse type of information and how to effectively serve the diverse set of end users. In traditional domains such as banking, insurance, and retail industries data warehousing has been successfully implemented to address this problem (inmon96). In such industries, the problem of how to design and implement data warehousing has been well researched over the years and is well understood. In nontraditional domains such as the Environmental and Earth sciences, the problem of applying data warehousing technology is complex and needs further study.

2. Challenges

Environmental data warehouse is an example of a spatial data warehouse. “Spatial Data Warehouse is defined as an integrated, subject-oriented, time-variant, and nonvolatile spatial data repository for data analysis and decision making [8].” A data warehouse may use one of the data models such as the star schema, fact constellation schema, snowflake schema or the multi-dimensional model. For example, in a star schema, the data warehouse contains a central table called the fact table, comprising of the keys of each dimension, and a table for each dimension. In a spatial data warehouse, the dimensions may include both spatial and non-spatial. Spatial Data warehouses pose many challenging requirements with respect to the design of the data model due to the nature of analytical operations and the nature of the views to be maintained by the spatial warehouse.

The first challenge is due to the multi-dimensional nature of each dimension itself. In a traditional data warehouse the various dimensions contributing to the warehouse data are simple in nature, each having different attributes. On the other hand, the different dimensions in a spatial data warehouse comprise of different types of data, each of which is multi-dimensional. The various raster images such as satellite downloads, images generated from these satellite images describing various parameters including land-use, water, temperature have multiple dimensions including the geographic extent and coordinates of the image, the time and date of its capture, and resolution. Other such examples include aerial photographs. The regional maps represented as vector data also have a temporal dimension as they change over time. The streaming data collected from various sensors placed at different geographic locations that sense temperature, air quality, atmospheric pressure, water quality, dissolved oxygen, mineral contents, salinity, again have both spatial and temporal dimensions. Other dimensions include demographic data, census data, traffic patterns, and many such as these.

The second challenge is due to the nature of the queries posed to the scientific warehouses. As the queries typically involve accessing multiple dimensions, each of which in itself is multi-dimensional. We illustrate this with the following examples:

Example 1: A user may want to look at the changes in the vegetation pattern over a certain region during the past 10 years, and see their effect on the regional maps over that time period. This involves layering the images representing the vegetation patterns with those of the maps whose time intervals of validity overlap, and then traverse along this temporal dimension with the overlaid image. In the traditional data warehouse sense, this amount to first constructing two data cubes along the time dimensions for each of the vegetation images and maps, and then fusing these two cubes into one. One may envision fusing of multiple cubes. For example, if the user also wants to observe the changes in the surface water, population, etc., due the changes in the vegetation pattern over the years, fusion of such multiple cubes is needed.

Example 2: Another user may want to simulate a fly-by over a certain region staring with a specific point and elevation, and traverse the region on a specific path with reducing elevation levels at a certain speed, and reaching a destination, effectively traversing a 3-dimensional trajectory. This query involves retrieving images that span adjacent regions that overlap the spatial trajectory, but with increasing resolution levels to simulate the effect of reduced elevation level. Another important aspect of serving such queries additionally requires controlling the speed at which they are displayed to match the desired velocity of the fly-by.

3. Spatial Database System Architecture

The ingestion, processing and storing of satellite images in CIMIC is done as shown in Figure 1. Images are downloaded from NOAA satellites with the Quorum HRPT antenna and receiver systems. Once a day the new raw image files are moved to oversized hard drives on a UNIX HP platform. At the same time, a new elements.dat file with ephemeris data is captured through the web and placed in the PC running the QTrack ingest software, which assures that images ingested later on will have updated orbital elements information and require less navigational correction.

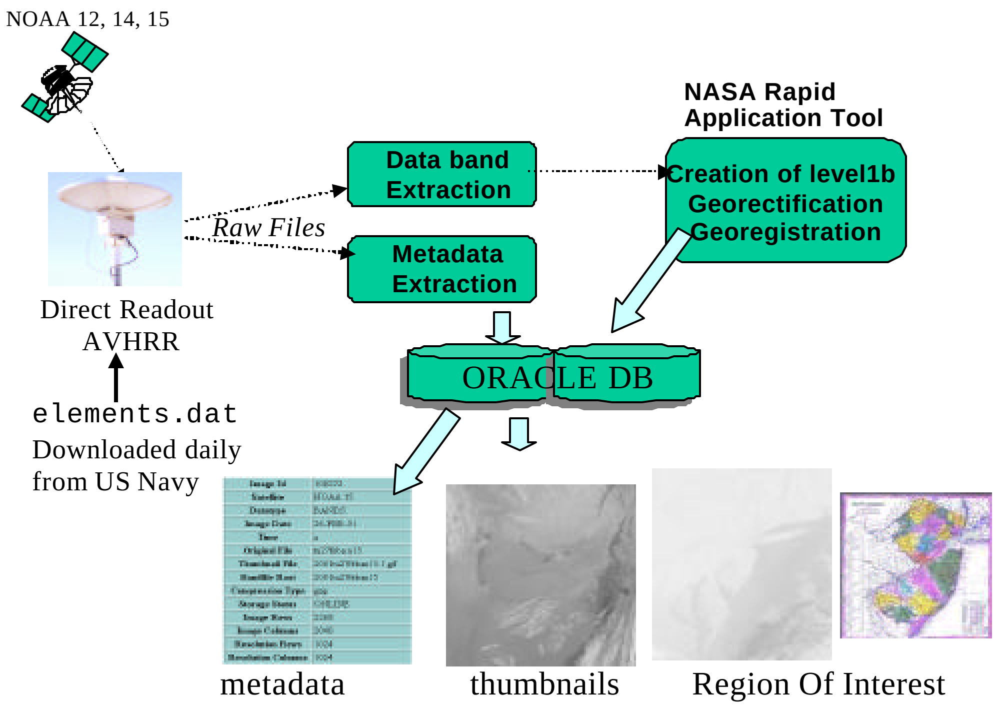

Figure 1: Preprocessing and Ingesting of Satellite Images

On the HP platform, raw files are fist classified by size. Files less than 20mb are automatically eliminated, and the remaining raw files are converted to level-1b by a quick-ingest routine, and then compressed. Level-1b files then go through the remap routine where images are clipped to a specific area of interest (New Jersey and surroundings) and projected to the Mercator projection. The resulting remap files are saved in an internal format (RAT format) and as bitmap files. These bitmap files are then classified using normalized regression routine, which employs a tool developed by NEC. Specifically, images with high regression coefficient (0.80 or greater) are classified as cloud free for the region of interest and flagged as so in the database. The RAT format files that emerge from the remap tool are used to create NDVI's. These NDVI's populate the database and become available to users through the web, and bi-weekly collection of NDVI's are made into a single NDVI images composite and are also available through the web. Due to the limited use of DBMS extenders for handling spatial data, we have implemented the database in two separate modules: One the relational DBMS to store metadata and thumbnail of images, and another a spatial data/flat file for images. Image files are tied with the DBMS by linking the image-id in the database with individual image files. The metadata of the images is maintained by an Oracle database through which image thumbnail images can be obtained. These images are indexed using an *SS-Tree* for enhancing the response time for the queries and insertions.

Interfaces are provided to querying the database based on time of capture, particular satellite or sensor instrument, type of image such as raw, composites, NDVI, water, temperature, etc. Essentially, users are provided with the image-ids, and the actual image is retrieved by clicking on the relevant image-id. Currently, it does not provide powerful capabilities to let users perform complex queries for advanced data analysis, such as trend or pattern analysis. In addition, no visual display tools are available to allow users to view image pattern changes over certain period of the range queries displayed with a speed specified by the users, nor capabilities to handle queries that simulate a fly-by over certain region as described in Examples 1 and 2.

Currently it uses ArcIMS from ESRI to process the image files (in .shp format), including layering the images, populating the metadata associated with the images, coloring, and composing fly-bys. These are then published on the web so that users can view them, zoom-in/out, move in different directions (north, south, east, west), or get associated metadata by clicking on a specific place. However, this is accomplished manually only for a pre-specified set of queries. Our goal is to accommodate ad-hoc queries by employing a data warehouse. As a result, for example, the above-mentioned fly-bys can be automatically generated upon users' request.

4. The Spatial Data Warehouse System Architecture

Our system comprises of a friendly geographic user interface, a powerful query processing engine that is capable of supporting various OLAP operations, an output rendering engine, and an spatial data warehouse, as shown in figure 2. Our data warehouse is based on the *cascaded star* model, described in section 6.

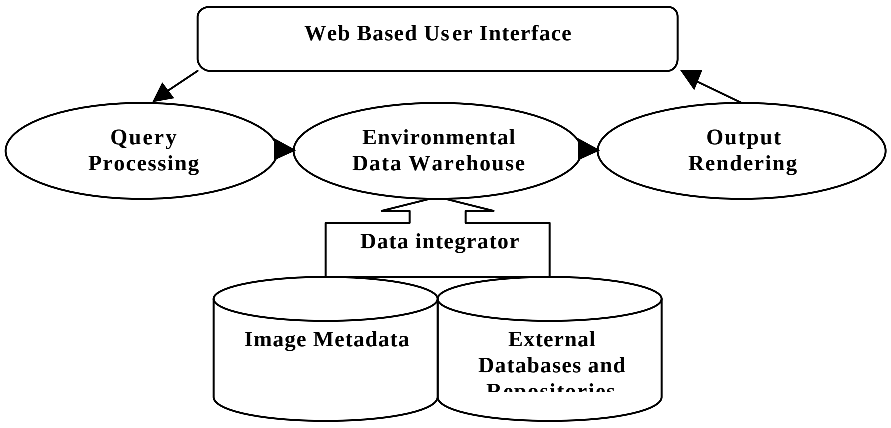

Figure 2: System Architecture

The data from different repositories, such as metadata databases, image database, databases of real-time streaming data from environmental sensors, etc., are first extracted, validated, transformed and then finally integrated, before loading into the warehouse. The data in the warehouse is periodically refreshed to reflect updates at the sources and purged from the warehouse, perhaps onto slower archival storage [10].

In general, the reason one builds a data warehouse is to construct data in a structured way and to allow pre-processing so that users can turn the data into useful knowledge quickly. Operational databases maintain state information, while data warehouses typically maintain historical information, and as a result, data warehouses tend to be very large and grow over time. Hence, the size of the data warehouse and the complexity of queries can cause queries process to take very long to complete, which is unacceptable in most decision support system environments. Also, a major performance challenge for implementing query processing and output representation is how we construct data warehouse in an efficient way.

4.1 Constructing an Efficient Data Warehouse

There are many ways to achieve data warehouse performance goals. Query optimizations and query evaluation technique can be enhanced to handle aggregations better, or using different indexing strategies like bit-mapped indexes and join indexes, etc. We consider implementing our GIS warehouse in the following two specific aspects to facilitate construction of the efficient data warehouse.

One commonly used technique is to selectively materialize/pre-compute frequently used queries. If we can do this pre-computation effectively and efficiently, then we can store many frequently accessed historical results in the data warehouse combined with different time periods, different resolutions, different aggregations, and different views, etc, at users' interests. In this way, the output processing can be achieved very fast, and sometimes automatically without any more computation efforts.

Firstly, let us look at the pre-computation for non-spatial data that are stored in RDBMS and are associated with spatial data. Picking the right set of queries to materialize is a nontrivial task. For example, we may want to materialize a query that is relatively infrequently used if it helps us answer many other queries quickly. We adopt the linear cost model from [8], where the data are stored in multi-dimensional data cubes, and each cell of the data cube is a view consisting of an aggregation of interest. The values of many of these cells are dependent on the values of other cells in the data cube. One common and powerful query optimization technique is to materialize some or all of these cells rather than compute them from raw data each time. A lattice framework is used to express dependencies among different cells in the total or partial order, and a greedy algorithm that works off this lattice determines a good set of cells to materialize [9]. We all know that dimensions of a data cube consist of more than one attribute, and the dimensions are organized as hierarchies of these attributes. For a simple example, the time dimension can be organized into the hierarchy: day, week, month, and year as follows:

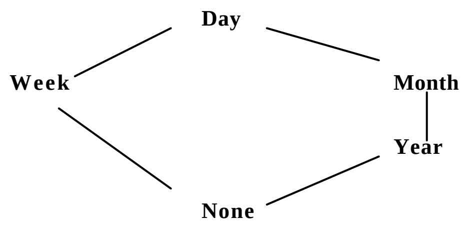

Figure 3: Sample Time Hierarchy

In the presence of above hierarchy, the dependency relationship is obviously seen. Consider a query that groups on the time dimension only, and we can have the following three queries possible: (day), (month), (year), each of which groups at a different granularity of the time dimension, also if we have total available for by month, we can use the results to compute the total grouped by year. Generally we selectively materialize the data cube based on query dependencies introduced by the conception of hierarchies.

Secondly, it is also essential to pre-process spatial data efficiently, which are more complicated than computing non-spatial data. For example, we may pre-process digital maps at different resolution levels and store them in the data warehouse, and users can combine them randomly to stimulate a fly-by, or pre-overlay the images representing the vegetation patterns with those of the maps having the same time intervals of validity, or pre-group a multi-color coded map to emphasize a particular category, or pre-interpolate spatial data over a large area which refers to the process of deriving elevation data for points where no data samples have been taken, etc. There is a big challenge for our project since our pre-processing is based on users' most frequent access interests that have to be updated frequently to meet changes.

Another challenge is that the above partial or total order relationship may not be suitable for spatial data dependency. For example, there is no dependency relationship among resolutions, and we can't compute high-level resolution based on low-level resolution or vice versa, or we can't overlay two images based on another overlaid image. Finding a dependency relationship among spatial data to avoid processing every raw image from scratch is our next step.

Another technique is to construct our data warehouse model in a different way that is an extension of the star schema, in which each dimension itself has a star schema of its own. We will explore this in detail in the following section.

4.2 The User Interface, Query Processing and Output Rendering Engines

A web based high-level user interface to a GIS must provide users with the necessary tools to store, retrieve, and analyze data so that they can perform their application-specific functions. More importantly, it is used to perform complex data analysis from the data warehouse without writing programs and should be comprehensive enough to let users get detailed analysis results and knowledge.

Moreover, after the translated SQL queries are processed in the data warehouse, an output will present multi-dimensional views of data to various front-end tools through different output processing engines. For example, OLAP servers can execute all OLAP operations, such as roll-up, drill-down, dicing and slicing, and generate results for data analysis and reporting, decision making strategies and advanced data mining. At the same time, users could require the data representation as the generation of a fly-by video with a trajectory, elevation and velocity.

When a spatial database is to be used interactively, graphical presentation of spatial data types (SDT) values in query results is essential. It is also important to enter SDT values to be used as "constants" in queries via a graphical input interface. The goal of querying is in general to obtain a "tailored" picture of the space represented in the database, which means that the information to be retrieved is often not the result of a single query but rather a combination of several queries. For example, in GIS application, the user may want to see a map built by graphically overlaying the results of several queries. Therefore, a user interface for output presentation should have at least two sub-windows: (1) a text window for displaying the textual representation of a collection of objects, containing the metadata or alphanumeric attributes of each spatial object, (2) a graphical window containing the overlay of the graphical representations of spatial data of several object classes or query results, which could be a generation of a fly-by video. We will consider implementing our system in this way in the near future.

The query engine translates the user inputs as SQL queries that will be inserted into data warehouses for further processing. The output representation engine is dealing with data representation using existing software such as PIT and IDRISI or newly developed applications. This part is mainly complicated by users' requirements because there are a lot of decision-support queries that are much more complex than OLTP queries and make heavy use of aggregation, and this is basically OLAP operations. Besides this, most users need some specific visualization results such as fly-by over a certain region staring with a specific point and elevation, and traverse the region on a specific path with reducing elevation levels at a certain speed, and reaching a destination, effectively traversing a 3-dimensional trajectory, or a fly-by over a certain time period for vegetation pattern change within New Jersey area, which is a process of image manipulation and representation.

5. Traditional Data Warehouse Models

A number of data models have been proposed to conceptually model the multi-dimensional data maintained in the warehouse. These include the star schema, the snowflake schema, and the fact constellation schema. Since our data model, the cascaded star model, is an extension of the star model, in the following, we present these three models with examples, and bring out the limitations of these models in representing the data in our spatial data warehouse.

5.1 The Star Schema

Perhaps, star schema, first introduced by Ralph Kimball, is the earliest schema used to model the data warehouse implemented as a relational databases. In this schema, the data warehouse contains a large central table (fact table) containing the bulk of data (dimensions) with no redundancy, and a set of smaller attendant tables (dimension tables) with one for each dimension. The schema graph resembles a starburst, with the dimension tables displayed in a radial pattern around the central fact table, as shown in Figure 4, where A is the fact table, and b, c, d, e and f are dimensions and represented by dimensional tables.

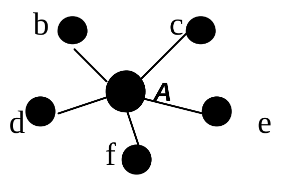

Figure 4: The Star Model

Note that in the star schema, only one dimension table represents each dimension, and each dimension table contains a set of attributes and joins with fact table by common keys when implemented as a relational database. Moreover, the attributes within a dimension table may form either a hierarchy (total order) or a lattice (partial order). Currently, most traditional data warehouses use a star schema to represent the multi-dimensional data model as it provides strong support for OLAP operations.

To illustrate, in the following, we provide an example of the implementation in star schema [8]. Suppose the multi-dimensional data for the weather in northeast region in USA consists of four dimensions: temperature, precipitation, time, and region_name, and three measures: region_map, area, and count, where region_map is a spatial measure which represents a collection of spatial pointers pointing to corresponding regions, area is a numerical measure which represents the sum of the total areas of the corresponding spatial objects, and count is a numerical measure which represents the total number of base regions accumulated in the corresponding cell.

The following figure illustrates the implementation for a star model in this case:

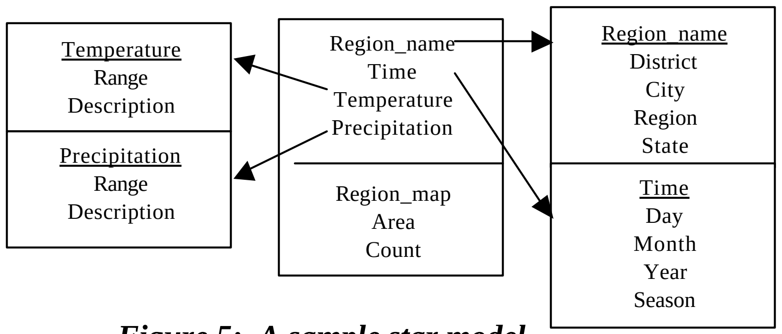

Figure 5: A sample star model

The following tables show some sample data set that maybe collected from a number of weather districts tested in northeast of USA.

Region_name	Time	Temperature	Precipitation	…
A111	02/23/01	33	1.4	…
B111	02/24/01	41	1.5	…
…	…	…	…	…

Region_name	District	City	Region	State
A111	A	Flushing	111	NY
B111	B	Edison	111	NJ
…	…	…	…	…

Time	Day	Month	Year	Season
02/23/01	23	February	2001	Winter
02/24/01	24	February	2001	Winter
…	…	…	…	…

Temperature	Range	Description
33	11	Chilly
41	12	Mild cold
…	…	…

Precipitation	Range	Description
1.4	21	Middle
1.5	22	Middle
…	…	…

From this sample, we can see that a star model consists of a fact table with multiple dimension tables, and the fact table joins the dimension tables with different keys. In this example, all attributes in each dimension table are only one-dimensional and can be expressed completely in one table. Our question is: if some or all of the attributes in the dimension tables are also multi-dimensional, i.e., one attribute in one dimension table has multiple attributes associated with it, how can we implement it in this model? The answer is impossible.

5.2 The Snowflake Schema

Snowflake schemas provide a refinement of star schemas where the dimensional hierarchy is explicitly represented by normalizing the dimension tables, and therefore further splitting the data into additional tables (see Figure 6). Such a table is easy to maintain and saves storage space because a large dimension table can become enormous when the dimensional structure is included as columns.

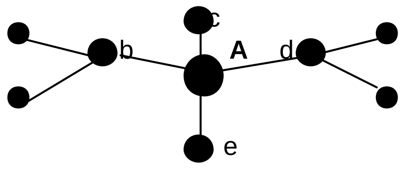

Figure 6: The Snowflake Model

However, only some dimensional tables are normalized and this normalization reduces the effectiveness of browsing since more joins will be needed to execute a query. When applied to spatial attributes for each dimension table in our case, it is obviously not well suited.

5.3 The Fact Constellation Schema

Sophisticated applications may require multiple fact tables to share dimension tables. The dimensions of this expanded star schema can be normalized into a snowflake schema. These multiple fact tables can separate the detail and the aggregated values instead of maintaining a single and huge fact table, which may speed the queries processing. See Figure 7 for this schema, where fact table A and B share the dimensions h and i.

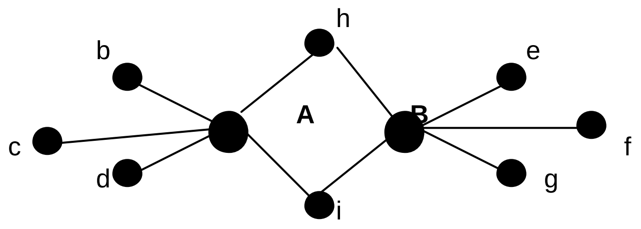

Figure 7: The Fact Constellation Model

However, there are some disadvantages of using the fact constellation schema. For example, for data warehouse with high cardinality, i.e. high number of hierarchy, numerous fact tables must be created, which increase the complexity of the design. Furthermore, for spatial oriented attributes for each dimension table, only one dimension table is not enough for holding the properties of each attribute.

6. The Cascaded Star Model

In this section, we present an outline of our spatial data warehouse model, called the **cascaded star schema**, which is an extension of the star schema, where each dimension itself has a star schema of its own. There are a number of research studies in the area of spatial data warehouses (see the reference list). The work proposed by Han et al. is closely related to our work. Han et al. [8,9] study the problems associated with the design and construction of spatial data cubes. It distinguishes the various dimensions in the spatial data warehouse as non-spatial, spatial-to-non-spatial, spatial-to-spatial, based on how they transform when that dimension is generalized. They provide how the various operations such as roll-up, drill-down, slicing and dicing, and pivot can be carried out. While we recognize that each spatial dimension in a data warehouse in itself is multi-dimensional and argue that the data warehouse model need to be enhanced to handle this. The cascades star schema is shown in Figure 8, where A is the fact table, and b, c, d, e and f are dimensions that are also multi-dimensional.

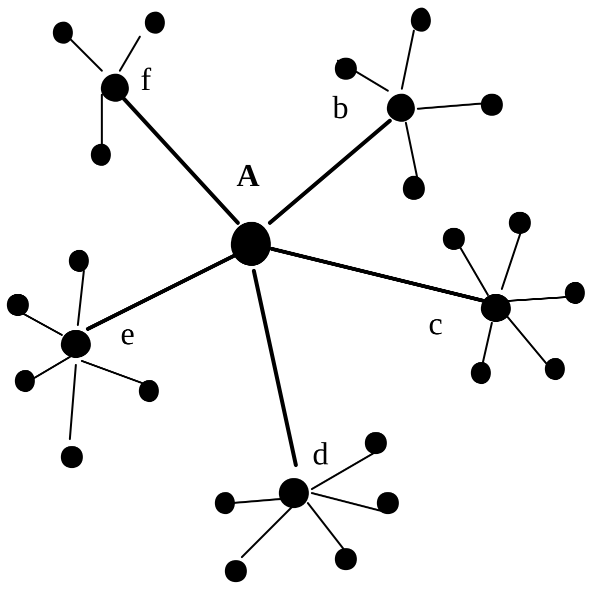

Figure 8: The Cascaded Star Model

The multi-dimensional nature of each dimension is illustrated with an example in figure 9. In here, the fact table comprises of the various dimensions of the spatial data, which include land-use, temperature, water and vector maps. As can be seen, each of these dimensions in turn is multi-dimensional, represented as a star. To illustrate, the land-use dimension comprises of a fact table of its own with dimensions time, spatial and attributes, where the time dimension is comprised of attributes year, date and time of capture of the image; the spatial dimension is comprised of the x, y coordinates of the lower left hand and corner and the upper right hand corner of the region covered by the image, and the resolution; the attributes dimension is comprised of the amount of vegetation, developed, barren, forested upland, etc. in the image. Similar to land-use, as can be seen from the figure, themes and water dimensions are also multi-dimensional in nature.

In the paper, we will present our detailed data model, and introduce the necessary primitives that enable the evaluation of different queries. We will also discuss what the different warehouse operations such as drill-down, roll-up, mean in the semantic sense in the cascaded star schema, and show how they can be carried out. We will present the architecture of our prototype and the guidelines for implementation.

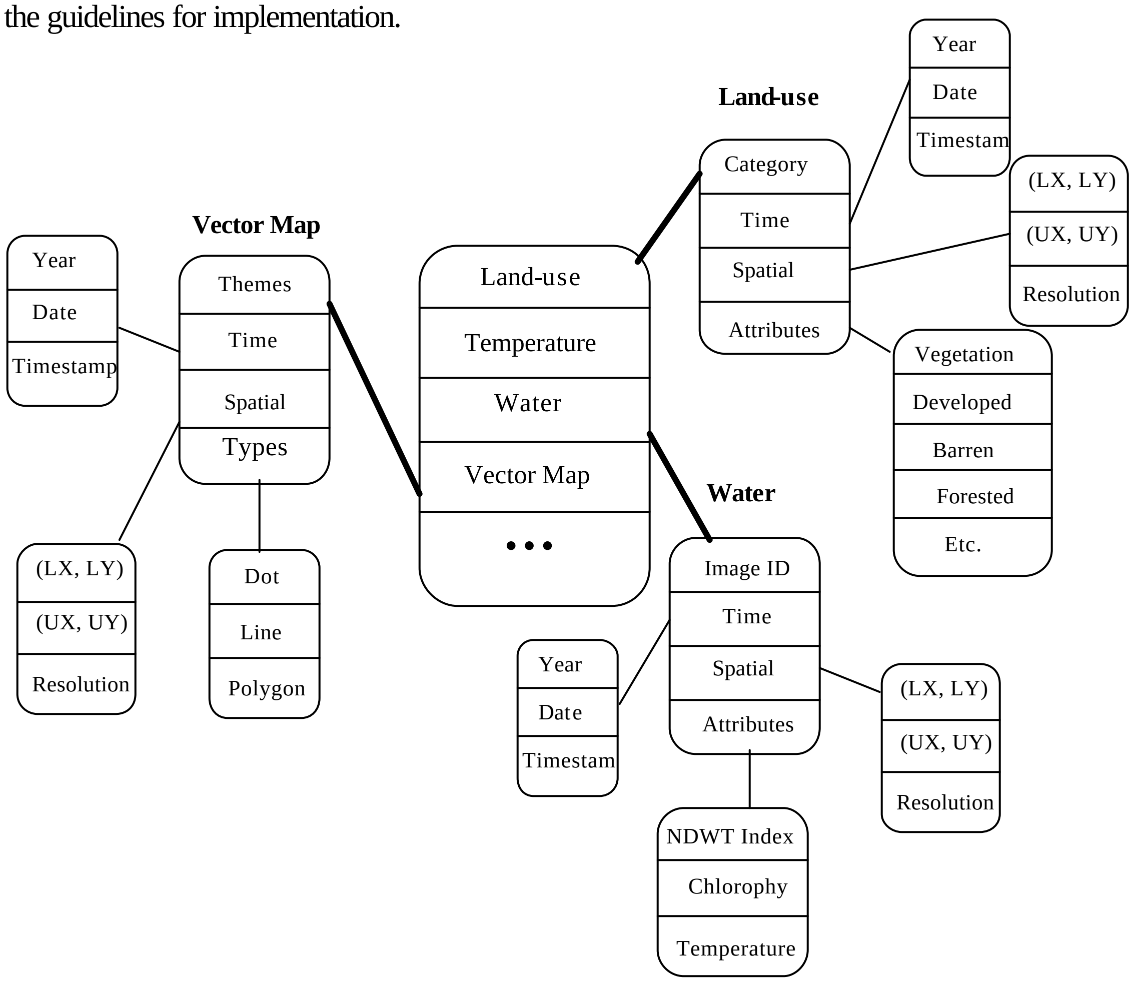

Figure 9: A Sample Cascaded Star Model

The following tables show some examples of these dimensions:

Fact table:

Land_use	Temperature	Water	Vector Map	…
Abc	44	221	111	…
…	…	…	…	…

One dimension table: "Vector Map"

Vector_map	Themes	Time	Spatial	Types
111	New Jersey	01	A	
…	…	…	…	…

Another dimension table for an attribute "Time" in "Vector Map":

Time	Year	Date	Timestamp
01	2000	3/23/00	12:00am
…	…	…	…

In the above example, we can see that a fact table is joined with several dimension tables as in the star model, and each attribute in the dimension tables is self multi-dimensional with another dimension table joined with it. In this easy way, we implement a cascaded star model for each multi-dimensional attribute in the dimension tables, which explicitly provides support for attribute hierarchies. However, the previous star schema cannot accomplish such multi-dimensional attribute structures in a single way.

We want to address the difference between a cascaded star model and a snowflake model. Someone may get the false impression at first sight that there is no big difference between these two models since they both have multiple extensions for some spatial dimensions. However, a snowflake model just normalizes some dimensions to reduce a big dimension table for easy maintenance and storage saving, whereas a cascaded star model claims each dimension itself is multi-dimensional by the nature.

6.1 OLAP Operations on the Cascaded Star Model

Now let us examine some popular OLAP operations, i.e., roll-up, and drill-down, slicing and dicing, and pivoting, and analyze how they are performed in the spatial data cube we constructed in a cascaded star model. OLAP are traditional data warehouse operations that provide users to view data from different perspectives, hence, OLAP support user-friendly environment for data analysis and prepare for advanced data mining process. In the system architecture we proposed, it is part of the output rendering engine.

These operations have been discussed intensively in the traditional data warehouse and spatial data cube in star model [7]. Our concentration is that how they can be efficiently operated in the star cascaded model with selectively materialization, which means aggregating and generalizing data from multi-dimensional attribute tables. Consider the example 1 we mentioned above. A user may want to look at the changes in the vegetation pattern over a certain region during the past 10 years, and see their effect on the regional maps over that time period. This query involves two very commonly used querying operations of OLAP: "drill-down" and "roll-up". We constructed the time hierarchy with a partial order in the above and they underlie these two operations. Drill-down is the process of viewing data at progressively more detailed levels, for example, a user drills down by first looking at the vegetation pattern per year and then comparing the vegetation pattern by specific month within different years. Roll-up is just the opposite, which is the process of viewing data in progressively less detail. In roll-up, a user starts with the vegetation pattern on a given month, then looks at the total pattern in that year, and finally, compares the patterns among 10 years. With selective pre-computation of certain data cells in the multi-dimensional data cube, such as vegetation pattern for each month within each year, we can easily process this query.

7. Related Work

Research in data warehousing is a relatively new area. In the following we review the research contributions as well as the prototypes that are most relevant to our work. Han et al. [8,9] proposes a spatial data warehouse model in which both spatial and non-spatial dimensions and measures exist. It proposes spatial data cube construction based on approximation and selective pre-computation spatial OLAP operations, such as merge of a number of spatially connected regions. The pre-computation involves spatial region merge, spatial map overlay, spatial join, and intersection between lines and regions.

Microsoft TerraServer [2] stores aerial, satellite, and topographic images of the earth in a database available via the Internet, where the users are provided intuitive spatial and text interfaces to the data. Basically terabytes of "Internet unfriendly" geo-spatial images are scrubbed and edited into hundreds of millions of "Internet friendly" image tiles and loaded into a data warehouse. The TerraServer adopts a "thin-client and fat-server" model, which consists of three tiers: the client tier, the application logic tier, and the database system tier. Users can search the data warehouse by coordinates and place names, and can easily view the images with different resolutions by simply clicking on it. The application logic responds to the HTTP requests and interacts with the back end database to fetch the results. The database is a SQL server 7.0 RDBMS containing all images and meta-data of images that are pre-processed and stored, for example, all levels of the image pyramid (7 is maximum) are pre-computed and stored. However, this system does not provide powerful and comprehensive image pre-processing tools such as spatial OLAP for advanced spatial data analysis. Moreover, the RDBMS integration with image repository has inherent problems, as SQL server 7 stores imagery in JPEG or GIF format which does not have much flexibility in handling spatial data.

However, none of the prior researchers recognize that each dimension in a data warehouse in itself is multi-dimensional. As a result, much of the work in spatial data warehousing is based on the star model. However, this work does not address the issue of the nature of spatial data warehouse.

8. Conclusions and Future Research

In this paper we focused on the problem of applying data warehousing technology in order to efficiently manage, store as well as effectively serve users of environmental and earth science information centers. An example of such centers is the Regional Application Center, which is collaboration between NASA, Rutgers CIMIC and New Jersey Meadowlands Commission (NJMC). In this paper, we recognize that environmental data warehouse differs from that of a traditional data warehouse in that, each dimension in itself is multi-dimensional in nature. We have proposed a new data model, called the cascaded star model to accommodate this. In this paper, we have provided a limited treatment to the OLAP operations. Our future work includes formalizing the necessary primitives that enable the specification and execution of queries, and the semantics of various warehouse operations including, drill-down and roll-up and the evaluation of these operations.

9. Acknowledgment

This work is supported in part by the Meadowlands Environmental Research Institute, Rutgers University.

References

[1] Nabil R. Adam, Aryya Gangopadhyay, “Database Issues in Geographic Information Systems”, Kluwer Academic Publishers, 1st edition, 1997.

[2] Tom Barclay, Jim Gray and Don Slutz, “Microsoft TerraServer: a spatial data warehouse,” Proceedings of the 2000 ACM SIGMOD on Management of data, pages 307-318.

[3] Peter Baumann, “Web-enabled Raster GIS Services for Large Image and Map Databases,” Proceedings of the ACM DEXA2001, pages 870 - 874.

[4] Wendolin Bosques, Ricardo Rodriguez, Angelica Rondon and Ramon Vasquez, "A Spatial Data Retrieval and Image Processing Expert System for the World Wide Web," 21st International Conference on Computers and Industrial Engineering, 1997, pages 433-436.

[5] Ron Briggs, “NSDI Demonstration Project: Final Report”, http://www.bruton.utdallas.edu/research/usgs/usgsframe.html

[6] Volker Coors, Volker Jung, “Using VRML as an Interface to the 3D Data Warehouse”, *Proceedings of the third symposium on Virtual reality modeling language*, 1998, Page 121-129.

[7] Martin Ester, Hans-Peter Kriegel, Jorg Sabder, “Knowledge Discovery in Spatial Databases”, Invited Paper at 23rd German Conf. on Artificial Intelligence (KI ’99), Bonn, Germany, 1999.

[8] Jiawei Han, Nebojsa Stefanovic, and Krzysztof Koperski, “Object-Based Selective Materialization for Efficient Implementation of Spatial Data Cubes ", IEEE Transactions on Knowledge and Data Engineering, 12(6): 938-958, 2000.

[9] J. Han, N. Stefanovic, and K. Koperski, “Selective Materialization: An Efficient Method for Spatial Data Cube Construction", Proc. 1998 Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD'98), Melbourne, Australia, April 1998, pp.144-158.

[10] Venky Harinarayan, Anand Rajaraman, and Jefferey D. Ullman, “Implementing Data Cubes Efficiently”, Proceedings of ACM SIGMOD Int'l. Conf. on Management of Data, Montreal, Canada, June 1996.

[11] R. Holowczak, N. Adam, F. Artigas, and I. Bora, "Data Warehousing for Environmental Digital Libraries." To appear in Communications of the ACM, 2002.

[12] N. Widmann, P. Baumann, “Towards Comprehensive Database Support for Geoscientific Raster Data,” Proceedings of ACM-GIS'97, Las Vegas/USA, November 1997.