The Universal Data Cube

Data Cubes in the Semantic Web

Curran Kelleher
University of Massachusetts
Lowell, Massachusetts
ckellehe@cs.uml.edu

Georges Grinstein
University of Massachusetts
Lowell, Massachusetts
grinstein@cs.uml.edu

Daniel Keim
University of Konstanz
Konstanz, Germany
keim@informatik.uni-konstanz.de

ABSTRACT

Data visualization and analysis tools often lack explicit support for hierarchical data cubes, their metadata, and dynamic merging of comparable data from multiple sources. The Universal Data Cube (UDC) is a vision for a world wide web in which richly annotated interlinked data cubes are first class citizens and rich web-based visualization and analysis tools are commonplace. The UDC vision requires many distinct components in order to function and flourish: the UDC Ontology provides the data model, the UDC Core Library is an API specification built around the model, the UDC SQL Library is an implementation of the model API based on RDBMS and RDF technologies, and UDC XML is an XML encoding of interlinked data cubes. These four parts form the basis of an open infrastructure upon which interoperable tools for data publishing, data navigation, interactive visualization and analysis can be built.

General Terms
Modeling

Keywords
OLAP, Data Cubes, Semantic Web, Service Oriented Architecture, Visual Analytics, Data Integration

1. INTRODUCTION

Users of visualization and analysis systems face many persistent problems. One of the biggest challenges faced is data preparation. Tools use many incompatible data models and formats, such as independent data tables, the entity-relational model (ERM), hierarchies only, or OLAP cubes. To further complicate the process, each model has a plethora of interchange formats, many of which are tool-specific. Each time one wants to use a given tool, one must first tailor the data to the tool, which may require tremendous effort and not be feasible for non-experts.

Some of this work was done during a stay at University of Konstanz.

Perhaps a larger problem is that of finding relevant data. When a domain expert wants to pose a question to the world of publicly available data, extensive research or a priori knowledge about data publishers is required. The inverse task of publishing data is also fraught with difficulty. If for example a small non-profit organization has collected data they would like to manage and make publicly available, they must choose (or create from scratch) a method of data organization, a method of data publication, and a set of formats to support.

Developers of visualization and analysis tools often must build their data infrastructure from scratch. This leads to much duplicated effort, and the data model of the resulting infrastructures is often driven by the needs of the specific functionality of the tool or dataset and is typically not usable by other applications.

Metadata support and interoperability is also a huge challenge throughout the processes of data collection, publishing, consumption, and analysis. What does the column named pop really mean? What is its unit of measurement? Why can’t my tool resolve automatically that record 5 in dataset A refers to the same object in the world as record 9 in dataset B? These are all metadata challenges.

The Universal Data Cube (UDC) seeks to address all of these issues. Conceptually, the UDC unifies the most common data models used by visualization and analysis tools, namely tables, hierarchies, and data cubes, into a single model. Unlike traditional OLAP metamodels, the UDC model does not assume that metadata (such as measure or record descriptors) is local to a particular data space or operational environment. The UDC model targets instead an open, service-oriented environment in which data and metadata publishing and consumption is distributed across many ownership domains.

The overall goal of the UDC is to provide a standard data and metadata framework for data publishing, navigation, interactive visualization and analysis. The design of the UDC is primarily driven by the needs of exploratory data visualization and analysis tools for the web of publicly available data, particularly regarding socioeconomic indicators. These needs became apparent during the development of WEAVE, a WEb-based Analysis and Visualization Environment. Though the UDC is motivated by issues in visualization, it is a general and widely applicable modeling solution.
Figure 1: The classes of the UDC conceptual model (ontology) and their relationships. Instances of Dataset, DataCube, DimensionInstance and MeasureInstance are defined local to a data provider and are backed by a database containing the contents of the data cubes. Instances of all other classes are defined globally, and can be referenced by many data providers.

which solves many long standing problems in distributed data and metadata management and integration.

2. RELATED WORK

The UDC is based heavily on concepts from the Semantic Web [1], in particular Linked Data [3]. The Resource Description Framework (RDF) [7] is the fundamental data model of the semantic web. It allows one to describe resources as a semantic network (a directed graph with labeled edges) in which each node and edge has its own URI (Universal Resource Identifier). RDF has a standard XML-based syntax called RDF/XML. Ontologies define shared vocabularies for RDF graphs containing class and property hierarchies. The Web Ontology Language (OWL) is a family of languages for defining ontologies in the semantic web. SPARQL is a query language for RDF graphs [9].

Linked Data is a set of RDF publishing guidelines proposed by Tim Berners-Lee which enables the Semantic Web to be browsable. The key Linked Data concept is that the URI of every RDF resource is a URL which, when accessed, provides a machine readable (typically RDF/XML) description of that resource. This is called a dereferenciable URI. When Linked Data descriptions contain references to other dereferencable URIs, a Linked Data network is formed. The public Linked Data network is referred to as the Linked Data Cloud.

The Semantic Web Client Library [2] is a Java library which exposes the entire Semantic Web to programmers as a single RDF graph. This is done by dereferencing HTTP URIs (traversing the Linked Data Cloud) and querying the Sindice Semantic Web index and search service.

Online analytical processing (OLAP) describes the analysis activities within a data warehousing operation [5]. A fundamental data structure used in OLAP is the OLAP cube, also known as a hierarchical data cube. An OLAP cube is a data structure for handling multidimensional hierarchical aggregations. In an OLAP cube, aggregation hierarchies are called dimensions, and aggregated numeric properties are called measures. Many standard operations are defined for OLAP cube views, such as roll up (increase aggregation level), drill down (decrease aggregation level), slice and dice (multidimensional selection), and pivot (rotating dimensional orientation).

A data cube may be stored in a relational database using a star schema, which contains data values in a central fact table and a dimension table for each dimension. Each tuple in the fact table contains pointers to members of the dimension hierarchies (one for each dimension) and a set of measure values. The dimension tables contain descriptors for members of the dimension hierarchies. The snowflake schema is an extension of the star schema which explicitly encodes dimension hierarchies using multiple related tables for each dimension. Multiple fact tables can share dimension tables. In this case the set of fact tables is called a fact constellation.

Several OLAP metamodels have been proposed which resemble the structure of the UDC ontology. Teiken and Flöring introduced the MUSTANG data analysis metamodel [13], which is based on the concept of domain specific modeling (DSM) and is oriented toward data integration and visualization. Sapia et. al. [10] proposed a conceptual extension to the entity-relationship model supporting OLAP cubes. The Common Warehouse Metamodel (CWM) [8] is an open industry standard for data and metadata integration which includes and OLAP metamodel. The CWM is based on a stack of Object Management Group (OMG) standards such as XMI, UML, and MOF.


Of the work discussed, none tightly integrates multidimensional modeling concepts with the semantic web or targets the open data community explicitly.

3. ONTOLOGY

Figure 1 shows a visual representation of the UDC ontology. Text represents classes. Branching lines represent one-to-many relationships, single lines represent one-to-one relationships.

A record represents a region of time, a region of space, an object category (set of objects) or an individual object. Example records include the year “1990” (a region of time), the country “USA” (a region of space), the industry sector “Min
A level contains a set of records which are on the same level in a record hierarchy. Example levels include “Year”, “Country”, “Industry Sector” and “Iris”. Levels can be hierarchical. Each level may have a parent level. For example, the parent record of the level “US state” may be the level “Country”. For a given level, the levels which have it as a parent are considered its child levels. A level tree represents a record topology. A path from the root to a leaf of a level tree (or any subpath thereof) represents a record hierarchy.

A level can have many parents from dimensions other than its own. This construct is used to represent “measures” (or, more precisely, sheafs) which evaluate to sets of records rather than numeric values. For example, the “US State” level of the “Space” dimension may be a parent level of the “Establishment” level of the industry dimension. This expresses the fact that US states (records of the level “US State”) contain businesses (records of the level “Establishments”), because businesses have geographic location.

A dimension contains a set of levels which could all potentially belong to the same level tree (and, by transitivity, the record topology within these levels). Example dimensions include “Time”, “Space”, “Industry” and “Iris Category”. All records are contained within levels, and all levels are contained within dimensions.

A quantity is a kind of numeric property. Example quantities include Currency, Quantity of People, Mass, and Speed. A unit is a concrete realization of a quantity. Example units include US Dollars, Thousands of People, Kilograms, and Kilometers per Hour. An aggregation operator defines a method of aggregating numeric values. Example aggregation operators include “Sum” and “Average”. A measure is a numeric property of records or products thereof, defined by a quantity and an aggregation operator. Here are some example measures:

<table>
<thead>
<tr>
<th>Name</th>
<th>Quantity</th>
<th>Aggregation Op.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Income</td>
<td>Currency</td>
<td>Average</td>
</tr>
<tr>
<td>Population</td>
<td>Number of People</td>
<td>Sum</td>
</tr>
<tr>
<td>Employment</td>
<td>Number of People</td>
<td>Sum</td>
</tr>
<tr>
<td>Speed Limit</td>
<td>Speed</td>
<td>Average</td>
</tr>
</tbody>
</table>

A dimension instance is a specific subset of records from one level of a dimension. A measure instance is a (measure, unit) pair. A data cube is structurally characterized by a set of dimension instances and a set of measure instances. The cells of a data cube are defined by all possible record products in which one record is taken from each of the data cube’s dimension instances (the cartesian product of all dimension instance record sets). The content of data cube is the mapping from its cells to numeric values for each of its measure instances.

A dataset is a collection of data cubes from the same data provider. A hierarchical data cube is a set of data cubes representing a given set of dimensions and measures. Hierarchical data cubes are of particular interest, as they support OLAP operations and are the structures that visual analysis tools will most likely target.

Additional structures containing unit conversion factors and mappings from identifiers in standard coding schemes (such as FIPS for geographic regions or NAICS for industry categories) to record instances are also part of the UDC framework. These structures support automatic unit conversion and lookup of records in existing data sets (identifier resolution).

4. OPERATIONS

Operations in the UDC model can be divided into those pertaining to knowledge (instances of ontology classes) and data (data cube content). Operations in the knowledge realm include create, read, update, and delete (CRUD) on instances of each class, and querying of the semantic network induced by a collection of class instances. Operations in the data realm include CRUD on data cube cells, data cube projection, unit harmonization and querying of hierarchical data cubes. Additional operations external to the UDC model can be identified pertaining to transformation of existing data into UDC model instances, such as identifier resolution.

4.1 Knowledge Operations

CRUD operations on dimensions, levels, records, measures, aggregation operators, quantities and units are self contained and atomic. CRUD operations on data sets, data cubes, dimension instances and measure instances must ensure consistency with the contents of the underlying data cube. For simplicity, we assume that the sets of dimension and measure instances of data cubes (in the knowledge realm) will not be modified when data cube content (in the data realm) is present.

In general, any semantic graph query such as those afforded by SPARQL [9] are possible on a UDC knowledge base. Example queries include “What are all the businesses in Massachusetts?” and “Show me all units available for expressing population”. A query type of particular interest is that which derives the data cubes within a given hierarchical data cube, for example “Show me all data cubes which include dimension instances representing the dimensions ‘Time’, ‘Space’, and ‘Industry’ and include measure instances representing the measures ‘Population’ and ‘Income’”. This type of query lays the foundation for executing hierarchical data cube queries which span both the knowl-
edge and data realms. Such queries may also span multiple data providers.

Merging of UDC knowledge bases to form composite knowledge bases is an operation fundamental to the high level goals of the UDC. This is equivalent to the problem of merging semantic graphs, which is readily possible by taking the union of object and property instances and resolving equivalent objects.

4.2 Data Operations

Once data cube metadata (instances of Dataset, DataCube, DimensionInstance and MeasureInstance in the knowledge realm) is established, insert, update and delete operations are possible on data cube cells. A cell insertion is specified by a cell location, defined by a set of records (exactly one from each dimension instance), and a cell value, defined by a set of (measure instance, numeric value) pairs (exactly one for each measure instance). A cell update is specified by a cell location and a partial cell value, defined by a set of (measure instance, numeric value) pairs which represent a subset of the cube’s measure instances. A cell deletion is specified by a cell location only and removes the cell entirely from the cube.

Data cubes may be projected. A data cube projection is defined by a set of cell locations and a set of measure instances. The result of a data cube projection is another data cube containing only the specified cells and measure instances. The units of the projection measure instances may not be the same as those used in the data cube’s measure instances. In this case, unit conversion must be performed using relevant unit conversion factors.

Since hierarchical data cubes are defined in a global manner, their data cubes can be dynamically assembled from many data providers based on a query of the UDC knowledge graph. At this higher level of abstraction, hierarchical data cubes may be queried. We have developed a query language for hierarchical data cubes based on the notion of a record selection. A record selection defines a subset of records within a dimension. A set of record selections (one for each dimension) and a set of (measure, unit) pairs (one for each desired measure) defines a hierarchical data cube query. The result of this query is also a hierarchical data cube.

Our query language allows compact representations of record selections. A record subhierarchy is defined by a (record, level, full) tuple where record defines a record at the root of the subtree, level defines the level of the leaf nodes of the subtree, and full is a boolean flag indicating whether the full subtree induced by record and level should be returned, or only its leaves (records in level which are ancestrally contained within record). Our query language allows composition of such hierarchy subtrees via union, intersection and relative compliment operators. This allows expressive queries such as: “Show me average income in US dollars for the construction industry in year 1995 for all US states and the counties of Texas.”

5. IMPLEMENTATION

We have created a Java library defining the specification of the UDC Model and its operations, called udc-core. Our design, inspired by the Jena Semantic Web library [4], defines the UDC model and operations in an object oriented and implementation agnostic way, allowing third party libraries to provide implementations at run time using the factory pattern.

Our SQL-backed implementation of the UDC Model is called udc-sql. In this implementation, we use an RDF triplstore implementation (Jena SDB) to store the knowledge realm of a UDC model (instances of classes shown in Figure 1) in a relational database. We provide our own implementation of the data realm. In our implementation, the contents of each data cube is stored in its own fact table, using a single URI lookup table in place of many dimension tables. This database layout can be thought of as a star schema using a fact constellation (one table per cube) with one shared dimension table whose dimensional descriptors refer to are nodes in a colocated RDF graph.

We have built an XML interchange format for collections of data cubes, called udc-xml. This format is a direct serialization of the udc-sql data realm schema: its header contains the singleton dimension table mapping URIs to integers, and its body contains the fact constellation. Cells (rows) of fact constellation tables are serialized as XML elements containing CSV strings for compactness. An XML schema for representing hierarchical data cube queries is also present in udc-xml.

Based on our udc-sql and udc-xml libraries, we have built a server for managing UDC models, called udc-server. This server exposes a RESTful web API whose methods mirror the structure of the udc-core API, using RDF/XML for knowledge interchange and udc-xml for data interchange.

Based on udc-server, we have implemented udc-client, which is an implementation of udc-core which acts as a proxy to a federation of udc-server instances. Our client makes use of the Semantic Web Client Library [2] for discovery and query of all public UDC knowledge bases in the Internet. The server federation which a client connects to may be the collection of all udc-server instances in the internet, i.e. the Universal Data Cube.

6. CONCLUSIONS

We introduced the Universal Data Cube (UDC) Ontology, which provides the conceptual foundation for describing semantically rich hierarchical data cubes. Operations and elements in the ontology are categorized into two realms: knowledge (common definitions and metadata) and data (data cube content). We introduced an implementation stack which realizes the UDC Ontology and associated operations. In our implementation, the knowledge realm of the UDC model is realized using existing Semantic Web technologies, while the data realm is realized using a custom variant of the star schema.

Taken as a whole, the UDC ontology and system implementation together provide the foundation for a World Wide Web in which hierarchical data cubes and their metadata are readily discoverable and consumable. This in turn lays
the foundation for the development of visual data analysis platforms which consume such data, and will have a tremendous impact on social scientists, planning agencies and anyone with an interest in publically available data.

Due to the distributed and composeable nature of UDC model instances, it is possible for client systems to query across many UDC servers, partially reconstructing the desired regions of the complete model locally and using it for visualization and analysis tasks.

We foresee many opportunities for the development of services relating to the UDC such as data hosting, archiving, indexing, curation, and presentation. We also hope that others will create many UDC-based systems over time, leading to a rich data interchange platform and community.

7. ACKNOWLEDGMENTS
We would like to acknowledge the Open Indicators Consortium, participants in the WEAVE project, Georges Grinstein and Daniel Keim for their support and enthusiasm for the development of the UDC.

8. REFERENCES