Joslyn, Cliff A and Gillen, David and Fernandes, R and Damante, M and Burke, J and Critchlow, T: (2008) "Hybrid Multidimensional Relational and Link Analytical Knowledge Discovery for Law Enforcement", in: 2008 IEEE Int. Conf. on Technologies for Homeland Security (HST 08), pp. 161-166, IEEE, Piscataway NJ, https://doi.org/10.1109/THS.2008.4534442

Hybrid Multidimensional Relational and Link Analytical Knowledge Discovery for Law Enforcement

Cliff Joslyn, David Gillen,
John Burke, Terence Critchlow
Pacific Northwest National Laboratory
Richland, Washington 99352
Email: {cjoslyn,david.gillen,
john.burke,terence.critchlow}@pnl.gov

Matt Damante
and Robert Fernandes
Lawrence Livermore National Laboratory
7000 East Ave., Livermore, California, 94551-9234
Email: {mdamante,rfernand}@llnl.gov

***Abstract*—The challenges facing the Department of Homeland Security (DHS) require not only multi-dimensional, but also multi-scale data analysis. In particular, the ability to seamlessly move from summary information, such as trends, into detailed analysis of individual entities, while critical for law enforcement, typically requires manually transferring information among multiple tools. Such time-consuming and error prone processes significantly hamper the analysts' ability to quickly explore data and identify threats. As part of a DHS Science and Technology effort, we have been developing and deploying for Immigration and Customs Enforcement the CubeLink system integrating information between relational data cubes and link analytical semantic graphs. In this paper we describe CubeLink in terms of the underlying components, their integration, and the formal mapping from multidimensional data analysis into link analysis. In so doing, we provide a formal basis for one particular form of automatic schema-ontology mapping from OLAP data cubes to semantic graphs databases, and point the way towards future "intelligent" OLAP data cubes equipped with meta-data about their dimensional typing.**

I. Introduction

Identifying and analyzing relevant information is a critical skill for any organization. This is especially true in the area of homeland security, where effectively understanding the available information has significant national security consequences. The challenges facing the Department of Homeland Security (DHS) require not only multi-dimensional but also multi-scale data analysis. Tasks such as reporting, trend analysis, and resource optimization require the flexible, graphical display of multiple different *ad hoc* projections aggregated over different views and subsets of the data. But tasks such as pursuing a particular case or finding subtle connections between suspects require viewing the particular details of persons and situations together with their connections.

Thus the ability to seamlessly move from summary information into detailed analysis of individual entities is critical. But information technologies appropriate for one kind of use may not be appropriate for another, forcing analysts whose needs span paradigms to expend significant time and energy moving between tools instead of focusing on the analysis. This type of analysis traditionally requires: first, examining summary or trend views using one tool to identify interested collections; second, manually transferring the selection criteria into a second tool to identify and extract the individual entities which comprise the collection; and finally manually transferring the individual information to a third tool which is used to determine if a potential threat is real or not. This time-consuming and error-prone process significantly hampers analysts' ability to quickly explore the data and identify threats, and hybrid environments are needed which can address multiple purposes elegantly and simultaneously.

As part of a Department of Homeland Security (DHS) Science and Technology (S&T) effort, the Generalized Data-Driven Analysis and Integration (GDDAI) Project has been developing and deploying such hybrid data analysis capabilities to DHS Immigration and Customs Enforcement (ICE). The goal is to provide a seamless integration of analysis environments, and allow the analyst to focus on understanding the data instead of the tools.

This paper describes CubeLink, an environment providing hybrid data analysis by linking two distinct technologies:

- **Multidimensional relational data cubes:** OnLine Analytical Processing (OLAP) technologies provide intuitive and graphical access to the massively complex set of possible summary views available in large relational (SQL) structured data repositories.
- **Link analysis in semantic graphs:** Semantic graph (SG) technology also provides an intuitive and graphical method, but now to flexibly explore particular data items in the context of their breadth of connections.

GDDAI is building a joint environment in which users can freely operate by first identifying groups of data items of interest as cells in OLAP cubes, and then viewing the details of those data items in an SG. While a simple task in principle, this actually represents a serious technical challenge. While very successful as individual and distinct technologies, OLAP and SGs are very different paradigms. And while there is

978-1-4244-1978-4/08/$25.00 ©2008 IEEE

active research in the database world in both schema alignment (between databases) and ontology alignment (between semantic data stores) [3], there are no methods to automatically link relational data schemata from OLAP environments to the ontological data typing methods of SGs.

Aside from their radically different display mechanisms, OLAP cubes implement large, uniform collections of data vectors with a high-dimensional, hierarchically-structured (rollup) space; while SGs implement a heterogeneous collection of two-dimensional data vectors as links in a graph. Our approach consists of:

- Building a mathematical mapping taking each high-dimensional OLAP vector into a collection of binary SG links, one for each attribute;
- Mapping scalar variables into data attributes of the SG;
- Finally representing explicit OLAP hierarchies as collections of distinct links in the SG.

Our environment is being implemented for deployment within the Compliance and Enforcement Unit (CEU) at ICE, linking together current capabilities in OLAP through SQLServer and Oracle-based OLAP cubes on the one hand, and the currently-deployed Government-Off-The-Shelf (GOTS) link analysis environment Everest on the other.

In this paper, we present our mathematical mapping from OLAP "count cubes" to SGs, and the current implementation using Everest and the ProClarity OLAP client, in the context of an overview of the two technologies. While we use the Internet Movie Database (IMDB) (`imdb.com`) as a model of the databases used within ICE's sensitive law enforcement areas, this development provides the formal basis for general automated schema to ontology alignment for these particular database forms. Additionally, in equipping our data cubes with meta-data typing information about their dimensional structure, we are pointing the way to future intelligent OLAP applications wherein OLAP cubes can be simply integrated with other capabilities, such as statistical and geo-temporal analysis methods also of interest to GDDAI.

II. Count-Based OLAP Data Cubes

OLAP [2] is a relational database technology providing users with rapid access to multiple summary, aggregated views of a single large database. OLAP views databases through a collection of major foci called **dimensions**, organized as possibly hierarchical collections of **members**. Constructing the Cartesian product of dimensions allows aggregation of quantities, called **measures**, over any collections of records projected through any subset of dimensions. These OLAP "data cubes" give flexible, visual reporting interfaces that are easily and rapidly driven by users.

OLAP arose in the financial arena, where measures are primarily numerical quantities such as money. But in our law enforcement applications, measures are typically the numbers of entities (people, organizations, events) having certain characteristics. Thus our approach rests on a specialized OLAP architecture we are calling **count cubes** built around **count star schemata**. Dimensions can come in three types:

- **Categorical:** Members are elements of an arbitrary set with no structure.
- **Scalar:** Members are numbers, with possibly artificial binning.
- **Hierarchical:** Members are elements within an explicit hierarchy.

Additionally, any categorical dimension can be an **entity dimension** if it lists items of interest, each with a unique **entity id**, which possess properties of the other dimensions.

Given a collection of dimensions, the central **fact table** consists of a Cartesian product of the primary keys of all the dimension tables, and thus holding one row for each combination of dimension values present in the underlying database. Given this structure, a measure is built consisting of the number (`distinct count`) of entities from an entity dimension possessing the values of the other dimensions.

A portion of the cube star schema for the IMDB is shown in Fig. 1. The dimensions are:

- **Movie:** The entity dimension.
- **Director:** A categorical dimension, holding the name of the movie's director.
- **Budget:** A scalar dimension holding the movie's budget in dollars, binned within $1M intervals.
- **Location:** A hierarchical dimension with levels of city, province (state), and country.

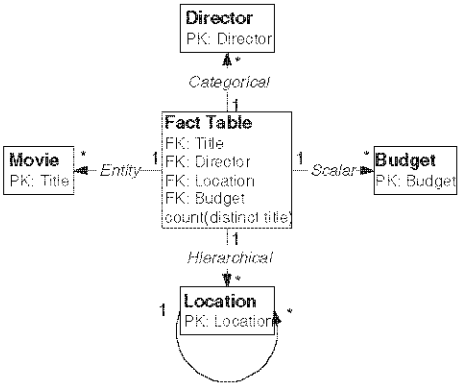
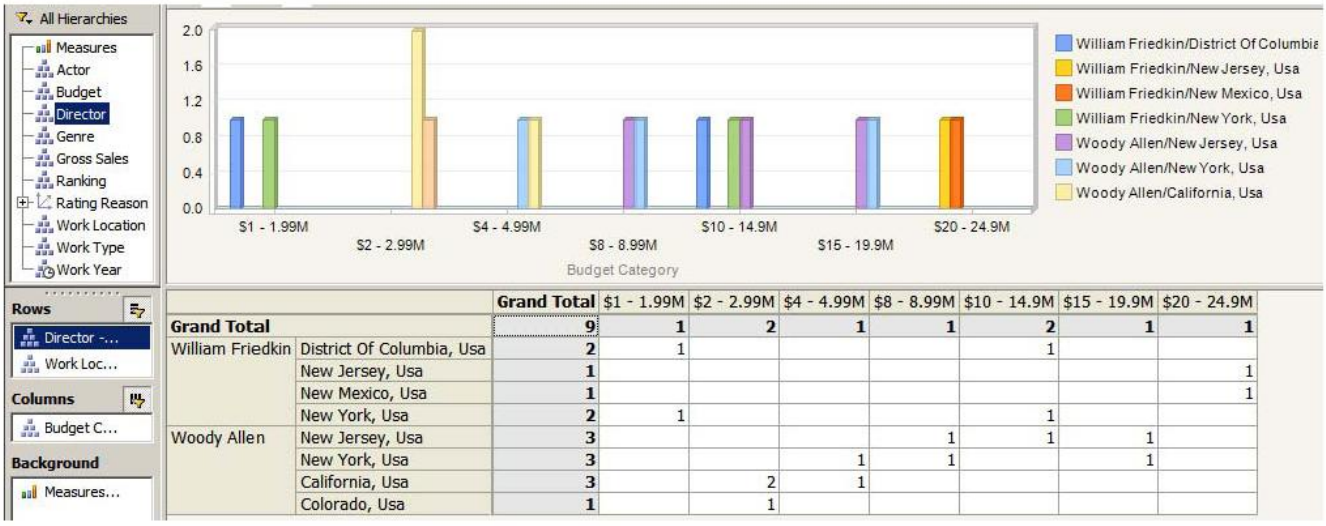


Fig. 1. A portion of a count star schema for the IMDB.

A portion of the fact table is shown in Fig. 2. A view onto the data cube in ProClarity is shown in Fig. 3, restricted to the two directors shown and to the years 1970-1985, and with the `location` dimension "rolled up" to the Province level. In our count cube methodology, each cell in the cube holds a count of the number of movies (entities) with that particular combination of dimensional values. Since the relation between movies and locations is one to many, this can result in multiple entries in the cube for each movie, as there are rows in the fact table. Thus the grand totals need not be the same as the sum of the cube values in a row, column, or page.

III. Everest Link Analysis of Ontologically Structured Semantic Graphs

SG databases [1] hold relational (predicate) data as a network of typed nodes connected by directed, typed links.



		Grand Total	$1 - 1.99M	$2 - 2.99M	$4 - 4.99M	$8 - 8.99M	$10 - 14.9M	$15 - 19.9M	$20 - 24.9M
Grand Total		**9**	**1**	**2**	**1**	**1**	**2**	**1**	**1**
William Friedkin	District Of Columbia, Usa	**2**	1				1		
	New Jersey, Usa	**1**							1
	New Mexico, Usa	**1**							1
	New York, Usa	**2**	1				1		
Woody Allen	New Jersey, Usa	**3**				1	1	1	
	New York, Usa	**3**			1	1		1	
	California, Usa	**3**		2	1				
	Colorado, Usa	**1**		1					

Fig. 3. OLAP cube view.

Movie	Director	Location	Budget
Annie Hall	Woody Allen	New York	$4M
Annie Hall	Woody Allen	California	$4M
Stardust Memories	Woody Allen	New Jersey	$10M
French Connection	William Friedkin	New York	$1.8M
French Connection	William Friedkin	DC	$1.8M

Fig. 2. A portion of a fact table.

Nodes maps to entities of interest, while links map to their relationships and properties. The collection of node and link types form an **ontology**, holding the composite typing information of the SG.

CubeLink uses the Everest SG tool which supports user-interactive visualization and sub-graph queries of SG ontologies and databases. Using Everest graphs, analysts can explore the SG, find or verify items of interest, and then optionally invoke additional tools for further analysis. By blending aspects of this SG with thematic and contextual analysis capabilities, analysts are equipped to identify previously hard-to-find and potentially unexpected relationships.

In order to map a count OLAP cube to an Everest SG, it will be asserted that there will exist a **sufficient ontology** for the count star schema meeting the following conditions:

- **Node Types:**
  1) There is one node type for each entity dimension, so that each entity maps to an **entity node** in the graph, with its identifier being the entity id.
  2) There is an attribute of the entity nodes for each scalar dimension, so that for each scalar dimension, each entity node has an attribute **scalar value** which is the member's value.
  3) There is one node type for each categorical dimension, so that each member may be represented as a **categorical node** in the graph, with its identifier being its value.
  4) Finally, there is one node type for each hierarchical dimension, so that each member *at any level* in the hierarchy may be represented as a **hierarchical node** in the graph, with its identifier being its value.
- **Link Types:**
  1) There is one link type for each categorical dimension, so that a **categorical link** connects an entity node to the corresponding categorical node.
  2) There is one link type for each hierarchical dimension, so that a **hierarchical value link** connects an entity node to the corresponding hierarchical node.
  3) There is one additional link type for each hierarchical dimension, so that a **hierarchical structure link** connects one hierarchical node to another according to the structure of the hierarchical dimension.

Fig. 4 shows a sufficient ontology for our schema, including:

- **Nodes:**
  - Entity node `Movie` with attribute `Budget`.
  - Categorical node `Person` for the director.
  - Hierarchical node `Location` within a hierarchy drilled down to `City`.
- **Links:**
  - Categorical link $\texttt{Movie} \xrightarrow{\texttt{Has\_Director}} \texttt{Person}$.
  - Hierarchical value link $\texttt{Movie} \xrightarrow{\texttt{Filmed\_In}} \texttt{Location}$.
  - Hierarchical structure link $\texttt{Location} \xrightarrow{\texttt{Has-Part}} \texttt{Location}$.

Note that a *sufficient* ontology is not a *necessary* ontology, since Fig. 4 includes node and link types not referenced in the cube. This is a critical part of the CubeLink methodology, allowing users to explore in detail relations in the database which are not available within the summary cubes.

IV. Contexts and Graphs

Along with Everest, CubeLink uses the ProClarity OLAP client tool. In ProClarity, users are able to perform several

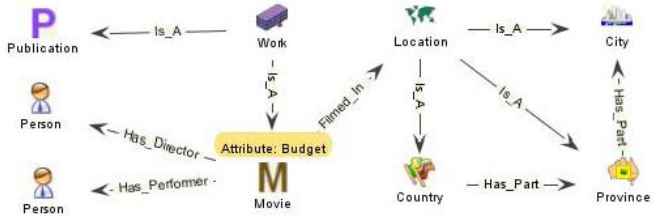


Fig. 4. A sufficient ontology.

classic OLAP operations such as drill down, drill from one hierarchy to another, pivot, filter, and summarize by any of the defined dimension hierarchies. OLAP query tools are typically best suited for analysis who have an interest in investigating anomalies and finding trends. When detailed investigation is needed, the user can right-click on a cell or cells to move into a detailed link-analysis of those cells contents.

Given a cube accessed in ProClarity, users can specify a **context** by identifying:

- **Rows:** Selection of certain members from certain dimensions along the rows.
- **Columns:** Selection of certain members from certain dimensions along the columns.
- **Filters:** Selection of certain selectional restrictions along certain dimensions, whether those dimensions are represented in the rows or columns or not.

Specification of such a context entails the identification of N distinct entities in the entity dimension. We then construct the SG, conforming to a sufficient ontology, of those entities by the following algorithm, providing thereby an adequate overall methodology for automatically aligning OLAP count star schema to SGs with sufficient ontologies [3].

- For each of the N entities entailed by the context:
  - Insert into the graph one entity node.
  - For each scalar dimension in the context, assign the appropriate scalar value to the entity node.
- For each categorical dimension in the context, whether on the rows or columns:
  - For each of that dimension's members in the context for which there is a positive measure, insert into the graph a categorical node with that member's value.
  - For each entity node in the graph, insert into the graph a categorical link connecting the entity to the corresponding categorical node for that dimension.
- For each hierarchical dimension in the context, whether on the rows or columns:
  - For each of that dimension's members in the context for which there is a positive measure
    - Insert into the graph a hierarchical node with that member's value.
    - If the hierarchical node is not a root of the hierarchy:
      - Insert a hierarchical node for its parent.
      - Insert a structure link connecting the hierarchical node to the parent hierarchical node.
      - Recur.
  - For each entity node in the graph, insert a hierarchical value link connecting the entity to the corresponding hierarchical node for that dimension.

We assume that node and link insertion has UNION semantics, so that duplicate nodes and links are either combined or not inserted again. Fig. 5 shows the SG generated by the context specified by the entire cube view shown in Fig. 3. All links are shown, including all ancestors of all `Locations` (in this case Provinces) in virtue of the inheritance of the hierarchical structure links.

V. Implementation

Implementation begins by parsing IMDB database files and loading them into an Oracle 10g relational database. The structure captures many facets of the database, including movie names, work locations, actor names, budget information, famous quotes, ratings, etc. The schema is composed of 46 tables. Some films are well represented with rich detail while other films may have only basic information captured.

A new OLAP database with a count star schema is then loaded from these relational structures. Our count star schema focuses on dimensions having relatively low cardinality and hierarchal support. Tables are often denormalized and heavily indexed as part of the transformation into more reporting friendly structures. The CubeLink star schema used in the ProClarity examples consists of eleven dimensional hierarchies, seen in the upper-left of Fig. 3, and one central fact table that counts distinct "works".

SQL Server 2005 Analysis Services is used to host the OLAP cubes. Hierarchies such as Country drilling down to Province and then to City are defined along with storage, user access, measure aggregations, and dimension organization settings. Terminology that the user understands is developed in an attempt to make the cubes as easy to navigate as possible.

ProClarity 6.3 is used to query the Analysis Services cubes. The API extensibility of ProClarity is unique and critical for extending the interface in support of the link analysis drillthrough. Users can specify a full context, including the dimensions on the rows and columns axes, any background filters applied, the measure being aggregated, and the actual values selected in the data grid. To generate the XML, we wrote an add-in to ProClarity using the ProClarity Software Development Kit 6.3 in C# using Microsoft's .NET framework. This adds a right-click context menu to the grid control, which iterates through the dimensions, does some minor data translation for scalar dimensions, then writes the XML file, and then launches Everest. Data translations were needed to expand the ranges in the scalar dimensions from their abbreviated human-readable formats like "$30-40M" to a machine-readable format that expresses `30000000 <= value < 40000000`.

Utilizing Everest's Relational Mapping capability, we mapped the IMDB schema to an ontology, creating a SG queriable by Everest. A metadata mapping file is required for

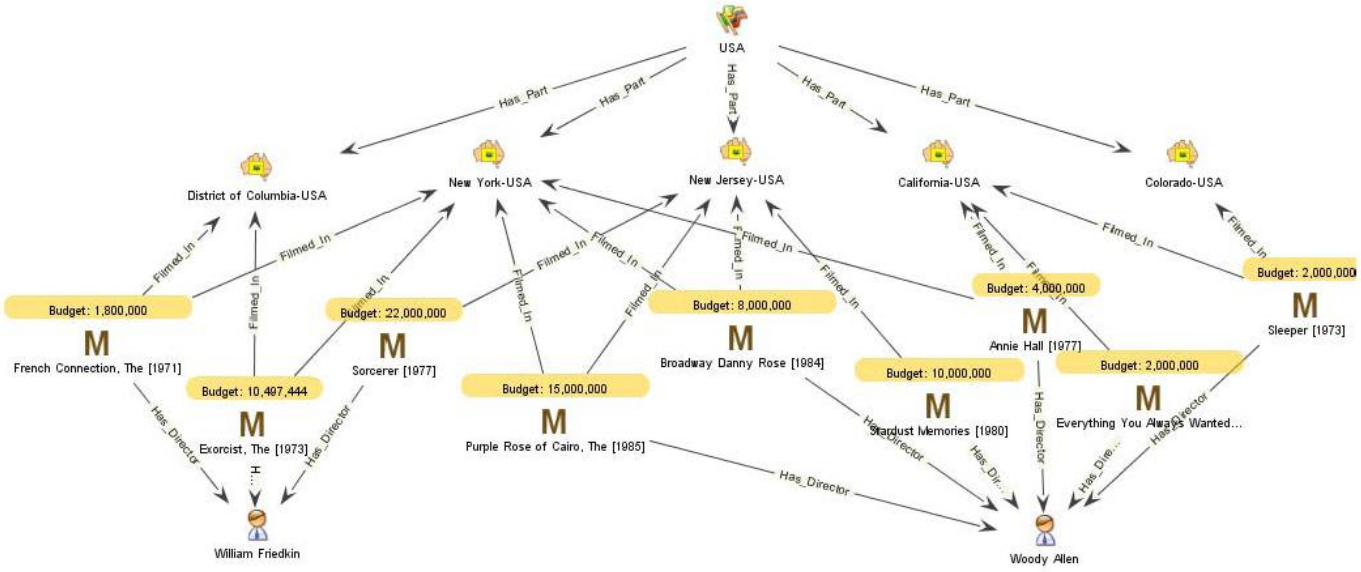


Fig. 5. Semantic graph.

Everest to know how count star schema dimensions map to ontology nodes. When Everest receives an XML formatted file, it begins building a set of graph pattern queries using the OLAP mapping metadata file to look up the types of dimensions and ontological concepts used by the cube selection query. Patterns are issued against the underlying persistence store returning all instances of subgraph pattern matches. Further analysis can then be performed on the resulting subgraphs to further analyze the entities backing the datacube selections.

VI. Mathematical Formulation

We conclude with a formal sepecification for the mapping from a count star schema with a single entity dimension to a nSG with a sufficient ontology.

Assume N (non-entity) dimensions $\mathcal{X} := \{X^i\}_{i=1}^N$. Each dimension is a set of **attribute values** $X^i := \left\{x^i_{j_i}\right\}_{j_i=1}^{M^i} \in \mathcal{X}$, so that $|X^i| = M^i$. Each dimension entails a set of **members** $P^i \supseteq X^i$ which live in a partially ordered set $\mathcal{P}^i := \langle P^i, \leq \rangle$. For two members $p^i_1, p^i_2 \in P^i$, we say $p^i_1 \prec p^i_2$ to mean that $p^i_1 \leq p^i_2$ and p^i_1 is an immediate child of p^i_2. We denote that an attribute value $x^i \in X^i$ belongs to a member $p^i \in P^i$ as $x^i \in p^i$ if $x^i \leq p^i$ (recalling that $\forall x^i \in X^i, \exists p^i \in P^i, x^i = p^i$).

In our example, we have $\mathcal{X} = \{\text{Director, Budget, Location}\}, N = 3$, so that we have three example variables prototypical of their classes:

- **Categorical Variable:** $X^1 = \text{Director} = \{\text{Director}_1, \ldots, \text{Director}_{M^1}\}$. X^1 has no hierarchy, so $P^1 = X^1$, and $\leq$ is empty.
- **Scalar Variable:** $X^2 = \text{Budget} = \{\$1M, \$2M, \ldots, \$25M\}, M^2 = 25$. $P^2 = \{[x, y) : x \leq y \in X^2\}$ is the set of all (half-open) intervals on X^2, e.g. $[\$3M, \$5M) \in P^2$, and $\leq$ is interval inclusion. Note that points like $[\$3M, \$3M]$ are in P^2, and we have that $\$3M \in [\$3M, \$5M)$.
- **Hierarchical Variable:** X^3 is location, a list of lots of cities, so that M^3 is large. So

$$P^3 = X^3 \cup \{\text{provinces}\} \cup \{\text{countries}\},$$

and $\leq$ is defined explicitly by the map, so that e.g. New York City (the member) $\leq$ New York State $\leq$ USA, and also New York City (the attribute value) $\in$ New York State.

So now think of two distinct **schemata:**

- **Data Schema:** $\vec{\mathbf{X}} := \bigtimes_{i=1}^{N} X^i$. This is the schema of the database, where the data records live, and each $\mathrm{x} \in \vec{\mathbf{X}}$ is called a **slot**. In our example, $\langle$Woody Allen, $\$4M$, New York City$\rangle$ is a slot.
- **OLAP Schema:** $\mathrm{P} := \bigtimes_{i=1}^{N} P^i$. This is the "frame" of the cube, the rows and columns in a view, and each $\mathrm{p} \in \mathrm{P}$ is called a **cell**. In our example, $\langle$Woody Allen, $[\$3M, \$5M)$, New York State$\rangle$ is a cell.

For a slot $\mathrm{x} \in \vec{\mathbf{X}}$ and cell $\mathrm{p} \in \mathrm{P}$ denote $\mathrm{x} \in \mathrm{p} := \forall 1 \leq i \leq N, \mathrm{x}_i \in \mathrm{p}_i$, where $\mathrm{x}_i, \mathrm{p}_i$ are the i'th component of the x, p vectors respectively, and of course $\langle$Woody Allen, $\$4M$, New York City$\rangle \in \langle$Woody Allen, $[\$3M, \$5M]$, New York State$\rangle$.

Now introduce a set of records of interest Z (e.g. people), forming the entity dimension, and two functions:

- **Attribute Function:** A function $f\colon Z \to \vec{\mathbf{X}}$ mapping each record to its slot (multidimensional attributes) in the data schema.
- **Drillthrough Function:** A function $d\colon \mathrm{P} \to 2^Z$ mapping each cell in the OLAP schema to the set of

records in that cell:

$$d(\mathbf{p}) = \{z \in Z : f(z) \in \mathbf{p}\}$$

- **Measure Function:** A function $m: Z \to \mathbb{R}$ mapping each record to a measure. Note that in the case of count cubes specifically, $m \equiv 1$.
- **Rollup Function:** A function $r: \mathbf{P} \to \mathbb{R}$ mapping each cell in the OLAP schema to the aggregates of the measures:

$$r(\mathbf{p}) := \bigoplus_{z \in d(\mathbf{p})} m(z)$$

where $\oplus$ is your favorite aggregation function, e.g. sum, average, max. In this case, r is **distinct count**.

We need only three more concepts:

- **Projections:** The full set of dimensions $\mathcal{X}$ and the full OLAP schema $\mathbf{P}$ represents the complete set of all available information. Let $K \subseteq \{1, 2, \ldots, N\}$ be a **projector**, which restricts our view to just the dimensions indexed by the $i \in K$. This gives us a number of additional mathematical objects:

$$\mathcal{X} \downarrow K := \{X^i\}_{i \in K}$$

$$\vec{\mathbf{X}} \downarrow K := \bigtimes_{i \in K} X^i, \quad \mathbf{x} \downarrow K \in \vec{\mathbf{X}} \downarrow K$$

$$\mathbf{P} \downarrow K := \bigtimes_{i \in K} P^i, \quad \mathbf{p} \downarrow K \in \mathbf{P} \downarrow K$$

$$f \downarrow K: Z \to \vec{\mathbf{X}} \downarrow K, \quad d \downarrow K: \mathbf{P} \downarrow K \to 2^Z,$$

$$(d \downarrow K)(\mathbf{p} \downarrow K) = \{z \in Z : (f \downarrow K)(z) \in \mathbf{p} \downarrow K\}$$

$$r \downarrow K: \mathbf{P} \downarrow K \to \mathbb{R}, \quad r(\mathbf{p} \downarrow K) := \bigoplus_{z \in (d \downarrow K)(\mathbf{p} \downarrow K)} m(z)$$

- **Filters:** $Y \subseteq Z$ be a **filter**, which restricts our view to just the records in Y. This gives us more objects:

$$d_Y: \mathbf{P} \to 2^Y, \quad d_Y(\mathbf{p}) = \{y \in Y : f(y) \in \mathbf{p}\}$$

$$r_Y: \mathbf{P} \to \mathbb{R}, \quad r_Y(\mathbf{p}) := \bigoplus_{y \in d_Y(\mathbf{p})} m(y)$$

- **Views:** Given a projector K and filter Y, then we have $\mathcal{X} \downarrow K, \vec{\mathbf{X}} \downarrow K, \mathbf{P} \downarrow K$ as above, and in addition now:

$$d_Y \downarrow K: \mathbf{P} \downarrow K \to 2^Y, \quad r_Y \downarrow K: \mathbf{P} \downarrow K \to \mathbb{R},$$

$$d_Y(\mathbf{p} \downarrow K) = \{y \in Y : f(y) \in \mathbf{p} \downarrow K\},$$

$$r_Y(\mathbf{p} \downarrow K) := \bigoplus_{y \in (d_Y \downarrow K)(\mathbf{p} \downarrow K)} m(y)$$

We now map these elements to components of an ontology.

- **Nodes:** We have the following node types V:
  - Records $z \in Z$, identified by their unique field combinations
  - Attribute values $x \in X$ of a categorical variable
  - Members $p \in P$ of a hierarchical variable
- **Attributes:** Attribute values $x \in X$ of a scalar variable
- **Links:** We have the following link types E:
  - Record attribute $z \xrightarrow{\texttt{Has\_Director}} x$
  - Record hierarchical value $z \xrightarrow{\texttt{Filmed\_In}} p$
  - Hierarchical values $p_1 \xrightarrow{\texttt{Has\_Part}} p_2$

We then have the following procedure for addin invocation:

- The user identifies a view by specifying a set of variables K and a set of filtered records Y
- The user identifies a collection of cells $\mathbf{Q} = \{\mathbf{q}\} \subseteq \mathbf{P} \downarrow K$ in that view
- The system creates the first set of nodes V_1 as the collection of data records corresponding to those cells

$$V_1 := \bigcup_{\mathbf{q} \in \mathbf{Q}} (d_Y \downarrow K)(\mathbf{q}) = \{y \in Y : \exists \mathbf{q} \in \mathbf{Q}, f(y) \in \mathbf{q}\}$$

- The system creates the second set of nodes V_2 as the collection of categorical attributes values

$$V_2 := \{f_1(v) : v \in V_1\}$$

where $f_1(v)$ is the first component of $f(v)$, that is, the eye color x^1 of the record v.

- The system creates the first set of links E_1 from records to their categorical attribute values

$$E_1 := \left\{v \xrightarrow{\texttt{Has\_Director}} f_1(v) : v \in V_1\right\}$$

- The system creates the third set of nodes V_3 as the collection of hierarchical attribute values

$$V_3 := \{p^3 \ni f_3(v) : v \in V, p^3 \in P^3\}$$

- The system creates the second set of links E_2 from records to their hierarchical attribute values

$$E_2 := \left\{v \xrightarrow{\texttt{Has\_Location}} f_3(v) : v \in V_3\right\}$$

- The system creates the third set of links E_3 from hierarchical attribute members to their ancestors via the recursive definition

$$E_3 := \left\{p_1 \xrightarrow{\texttt{Has\_Part}} p_2 : p_1 \prec p_2 \text{ and } \left(p_1 \in V_3 \text{ or } \exists p \xrightarrow{\texttt{Has\_Part}} p_1 \in E_3\right)\right\}$$

- The system creates the graph as $\langle V, E\rangle = \langle V_1 \cup V_2 \cup V_3, E_1 \cup E_2 \cup E_3\rangle$.

References

[1] Barthelemy M, E Chow, and T Eliasi-Rad: (2005) "Knowledge Representation Issues in Semantic Graphs for Relationship Detection", Proc. 2005 AAAI Spring Symposium on AI Technologies for Homeland Security, AAAI Press, Stanford, pp. 91-98.

[2] Chaudhuri, Surajit and Umeshwar Dayal: (1997) "An overview of data warehousing and OLAP technology", *ACM SIGMOD Record*, v. 26:1, pp. 65 - 74

[3] Euzenat, Jérôme and Shvaiko, P: (2007) *Ontology Matching*, Springer-Verlag, Hiedelberg