NetIntel: A Database for Manipulation of Rich Social Network Data

Maksim Tsvetovat, Jana Diesner, Kathleen M. Carley

March 3, 2005
CMU-ISRI-04-135

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

There is a pressing need to automatically collect data on social systems as rich network data, analyze such systems to find hidden relations and groups, prune the datasets to locate regions of interest, locate key actors, characterize the structure, locate points of vulnerability, and simulate change in a system as it evolves naturally or in response to strategic interventions over time or under certain impacts, including modification of data. To meet this challenge, we need to develop a new mechanism for storing and manipulating social structure data.

Social structure data will be stored in a relational database capable of manipulating large quantities of data. The database is structured to preserve the character and integrity of the data in an extensible manner, and is extended with a number of functions specifically designed for manipulating graph-based data.

Contacts: maksim@cs.cmu.edu, jdiesner@cs.cmu.edu, kathleen.carley@cmu.edu

This work was supported in part by the DOD, the National Science Foundation under the IGERT program for training and research in CASOS, NSF ITR 1040059 and the Office of Naval Research N00014-02-1-0973. Additional support was provided by CASOS - the center for Computational Analysis of Social and Organizational Systems at Carnegie Mellon University. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the DOD, the Office of Naval Research, the National Science Foundation, or the U.S. government.

Keywords: Database, Social Networks

Table of Contents

List of Figures

1 Rationale for a Social Network Database

Social network analysis focuses on the relations among and between entities in a social or organizational system. For most of its history, social network analysis has operated on a notion of a *dataset* - a clearly delimited and pruned set of observations that have been encoded and parameterized using a particular set of assumptions or policies. In such a dataset the entities are represented as nodes, and the relations between them as edges or links. The datasets were often painstakingly collected by hand over long periods of time and pruned to illustrate a phenomenon or hypothesis with the greatest possible clarity. Thus, traditional SNA datasets came to be viewed as self-contained units of data, potentially with some shared characteristics (such as data collection or encoding methods) but also with potential for vastly different assumptions at any stage of the process.

Each dataset, as defined in the SNA terms, represents a largely self-contained conceptual chunk: a snapshot of a social system at one point in time. As one proceeds with analysis, new measures computed on the dataset are introduced, again largely self-contained but, except in a researcher's mind, still unconnected with the original data in a except in assorted ad-hoc fashions. Over the past ten years there have been a growing number of studies in which the researchers made use of the social network and attributes or networks at two or more points in time.

Social network data can be multi-mode (various types of relationships such as friendship, kinship), multi-link (connections across various meta-matrix entities) and multi-time period. Furthermore, nodes and edges can have multiple attributes such as the formal position of an employee in a company or the types of relationships between employees (multi-mode). We refer to data that is multi-mode, multi-link and multi-time period in which both nodes and edges can have attributes that carry information on to how to interpret, evolve, and impact these nodes and edges as "rich" data.

With the advent of tools for automated gathering of relational data such as AutoMap[3][4] and computational models including multi-agent social-network simulation tools such as Construct[9] and NetWatch[15], the sheer quantity of available data has increased exponentially. Moreover, most of the data gathered, processed, or simulated in this manner is rich network data of groups with hundreds or thousands of actors.

Ad-hoc approaches to storage and manipulation of such data significantly increase the labor burden upon the researcher, and it became apparent that new ways to query, manipulate and extract subsets of the data were required. Furthermore, as research groups in the field joined in large-scale projects, there arose a need for a well-defined data interchange format.

There is a pressing need to automatically collect data on social systems as rich network data, analyze such systems to find hidden relations and groups, prune the datasets to locate regions of interest, locate key actors, characterize the structure, locate points of vulnerability, compare and contrast alternative networks, visualize the structure of a system as a whole or in part and simulate

change in a system as it evolves naturally or in response to strategic interventions over time or under certain impacts, including modification of data. To meet this challenge, we need to move beyond the traditional approach[5].

The amount and quality of data collected by the automated data gathering tools and simulation tools suggest that a relational database (RDBMS) tool needs to be used to manage and query the data. However, dealing with large quantities of network data presents a number of unique challenges from the data warehousing point of view.

We discuss the design and construction of NetIntel - an RDBMS-based system for warehousing, merging and manipulating large network datasets.

2 Databases and Gathering of Network Intelligence

In the aftermath of the September 11th attacks, it was noted that coherent information sources on terrorism and terrorist groups were not available to researchers[10]. Information was either available in fragmentary form, not allowing comparison studies across incidents, groups or tactics, or made available in written articles - which are not readily suitable for quantitative analysis of terrorist networks. Data collected by intelligence and law-enforcement agencies, while potentially better organized, is largely not available to the research community due to restrictions in distribution of sensitive information.

To counter the information scarcity, a number of institutions developed unified database services that collected and made available publicly accessible information on terrorist organizations. This information is largely collected from open source media, such as newspaper and magazine articles, and other mass media sources.

Such open-source databases include:

- RAND Terrorism Chronology Database[8] - including international terror incidents between 1968 and 1997
- RAND-MIPT (Memorial Institute for Prevention of Terrorism) Terrorism Incident Database[11], including domestic and international terrorist incidents from 1998 to the present
- MIPT Indictment Database[18] - Terrorist indictments in the United States since 1978.

  Both RAND and MIPT databases rely on publicly available information from reputable information sources, such as newspapers, radio and television.
- IntelCenter Database (ICD)[7] includes information on terrorist incidents, groups and individuals collected from public sources, including not only traditional media outlets and public information (such as indictments), but also information learned from Middle East-based news wire services.

Separately, IntelCenter also collects information from Arabic chat-rooms and Internet-based publications - although value of such data is questionable and data may be tainted by propaganda.

The focus of these databases is the agglomeration of publicly available data and dissemination of it to researchers, both in the public and private sectors. Little of the work in large public databases has been focused on enabling social network analysis or link analysis of covert and terrorist networks. The IntelCenter Corporation has released a dataset mapping relationships between members of Al-Qaida[12]. However, that dataset was delivered as virtually a read-only diagram that did not facilitate quantitative analysis of the data.

Furthermore, the data in the above databases is frequently presented in a proprietary format, making it difficult to employ other software for analysis purposes.

On the commercial software frontier, I2 Corporation has been marketing Analyst's Notebook[6], a software product for integrating and charting network-based intelligence on criminal and terrorist organizations. This software is in use in many governmental and law enforcement agencies, and allows significant integration with data collection, communication and other technologies. A separate product, *iBase*, provides shared storage and data integration facilities of the product. However, the product implements very few quantitative analysis tools and does not allow ready export of network data into analysis packages.

The goal of the NetIntel project, as described in this chapter, is to provide a means for ready collection and integration of network data, with an emphasis on making the data available for quantitative analysis with standard software tools, and making the database accessible on the basis of open standards for database connectivity.

3 Existing Work

Storing and manipulating massive persistent graph data is a non-trivial preposition. Despite the fact that majority of data captured by businesses and organizations is relational in nature and can be efficiently described in terms of graphs, much of database and data mining research in the past decade has concentrated on propositional data[17].

In propositional data, instances and objects are assumed to be identical and independently distributed (*i.i.d.*). Relational data violates this assumption. Relationships among objects reflect dependence among instances, and the instances themselves are heterogenous. Rich social network data, such as information extracted via text analysis, further supports this fact by attaching semantics and attribute sets to both instances and the relations themselves.

Further, even the part of the data mining community that routinely deals with relational data has focused on learning patterns from the data and structure of relations. An important, but oft-overlooked aspect of storing relational data is that of data selection and transformation.

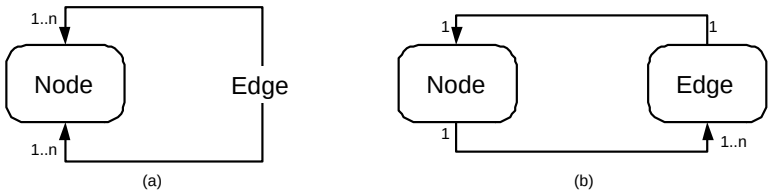

Figure 1: Schemata for Graph Data: (a)Simple relations, (b)Rich relational data

The basic schemata for storing graph data in a relational database (RDBMS) (figure 1(a)) is a many-to-many relation recursively defined on a table. Figure 1(b) illustrates a slightly more complicated scenario, turning edges from a simple relation to an object that can contain rich attribute sets.

To extract subsets of a large graph both of these schemata in a query language such as SQL, it is necessary to build a recursive **JOIN** operator. However, it should be noted that SQL[16] explicitly disallow recursive **JOIN**s, and thus lack facilities for ready implementation of graph data manipulation.

A number of research systems, such as Lorel[2] and QGraph[14], solve this problem by building graph-based DBMS systems from scratch or building on top of object-oriented storage mechanisms, and adding a graph query language.

Lorel[2] language, while not specifically designed for manipulation of social network data, is built using a graph metaphor as the underlying data model and thus suitable for storage of relational data. It also introduces existential quantifiers that allows selection of graph subsets based on existence (as opposed to properties) of their relations. However, Lorel can be only implemented on top of an experimental *Lore*[13] DBMS - which is no longer maintained and thus not suitable for use in a production system.

QGraph[14] presents a visual language specifically designed for querying and updating graph databases. A key feature of QGraph is that the user can draw a query consisting of vertices and edges with specified relations between their attributes. The response will be the collection of all subgraphs that have the desired pattern. QGraph implements an advanced set of quantifiers that allows the user to specify not only existence of relations, but also the cardinality of edge subsets, and also place conditional operators on both nodes and edges. QGraph is well-suited for dealing with general graph data, and can be well adapted for storing and retrieving complex social network data.

However, the graphical nature of the language that is one of its selling features for a stand-alone DBMS, presents a significant disadvantage for integration of QGraph into an analysis toolchain. QGraph also shares Lorel's disadvantage of being built on top of an experimental storage engine.

4 Storage Requirements for Social Structure Data

Mindful of the limitations of both traditional RDBMS systems and experimental graph query languages, we suggest that a scalable solution to storage and manipulation of graph-based data is not via creation of custom database systems or query languages, but rather via extension of database tools found and widely used in the industry.

The structure of the database shall be defined in a way that preserves the character and integrity of the data (i.e. is aware of underlying graph properties of the data). The structure shall be designed in an extensible manner, allowing easy addition of new attributes, node and edge types.

The database system shall not only keep track of the units of social structure data (such as nodes and edges) but also sources of such, enabling the creation of large-scale multi-source datasets while preserving the original data sources.

The database shall have an easy-to-use web-based interface, allowing users to enter, search and edit data as well as access manipulation and query capabilities described below.

5 Requirements for Data Manipulation Tools

The data manipulation tools shall be closely coupled to the database system described above. The foremost requirement for the subsystem is the ability to extract subsets of the data based on:

- the source (or sources) of data (e.g. "Find all social structure data that came from New York Times" or "Find all data that came from New York Times article from 10/10/2003")
- attributes of nodes and edges (e.g. "What is the network of people who were born in Syria?")

The manipulation tools shall include both existential, universal and cardinality quantifiers for specifying structure of subgraphs.

The manipulation tools shall be able to extract subsets of the network based on graph-theoretic properties of the network such as graph distance (e.g. "Find all nodes at a graph distance of 2 or less from a given node") and graph density (e.g. "Find all nodes embedded in subgraphs with given density").

The manipulation tools shall allow easy completion of incomplete datasets (e.g. "Given a set of people, find all organizations and resources connected to them")

The query tools shall enable the creation of time-slices from the complete dataset of any subset thereof, if time-dependent data is present.

Finally, the query tools shall be easily combined into scripts, resulting in extremely powerful structure-aware data manipulation capability.

6 Database Design

A DBMS chosen for the underlying storage engine must feature advanced query capabilities (both SQL and procedural), as well as stored procedure and trigger capabilities. For the current implementation, we have chosen PostgreSQL[1] database. Of the databases available under the GPL license, it offers the most complete implementation of ANSI SQL, and an implementation of PL/SQL - a procedural language that is portable to other industry database systems such as Oracle.

The majority of the data manipulation tools are implemented as functional extension to SQL and thus available within standard SQL queries. Two end-user interfaces to the database (a command-line system and a web-based interface) are also awailable. These interface serve as a means to import raw data from data gathering tools such as AutoMap and export data into analysis tools such as ORA.

The Web-based interface allows easy navigation and editing of large bodies of data, as well as some access to data manipulation and query tools.

6.1 Database Schema

The database schema is designed to preserve flexibility inherent in the source data while enforcing some regularity upon the datasets.

2 tables, **Node** and **Edge**, compose the basic graph structure. Two separate tables contains a set of Node Types and Edge Types, thus making the graph structure in the database semantically extendable.

A **Document** table stores data sources that contribute to the creation of the database and links them through many-to-many relations to the graph entities (Nodes and Edges).

A set of tables store domain-dependent data on each of the node types. These tables are not static in the database schema, but rather created automatically at the same time as a new Node Type, thus ensuring both flexibility and referential integrity of data.

6.2 Thesaurus

Due to the fact that data for the database comes from many disparate sources and includes many foreign names, alternative spellings of such names are inevitable.

The database uses a separate **Thesaurus** table to store alternative spellings of names of entities. When an entity (Node or Edge) is inserted, queried or updated, a *Trigger Function* checks spelling of the entity's name or ID and makes sure that the ID is spelled in a canonical way within the dataset.

Unfortunately, the data populating the Thesaurus table had to be compiled by hand. However, with a simple conversion tool, NetIntel can make use of thesauri written for use with AutoMap and can therefore capitalize on the manual work that was invested in their creation.

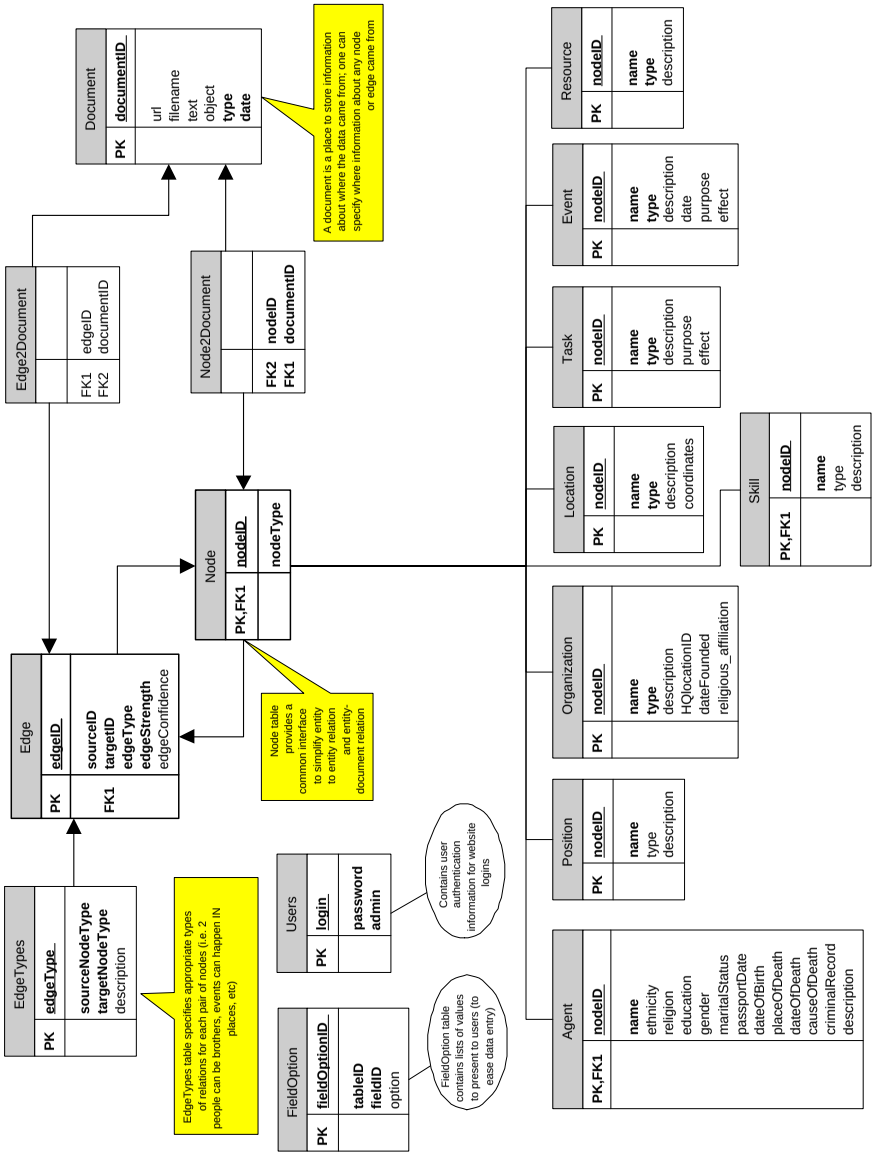

Figure 2: NetIntel Database Schema

7 Data Manipulation

The data manipulation tools are closely coupled to the database system described above. The foremost requirement for the subsystem is the ability to extract subsets of the data based on:

- the source (or sources) of data (e.g. "Find all social structure data that came from New York Times" or "Find all data that came from New York Times article from 10/10/2003").
- attributes of nodes and edges (e.g. "What is the network of people who were born in Syria?").

The query tools enable the creation of time-slices from the complete dataset of any subset thereof if time-dependent data is present.

The above queries are easily accomplished using SQL and the Document tracking tables. This query has been implemented as a part of **db2dynetml** export program and is useable with a single command-line option.

7.1 Graphs in NetIntel Database

Let the graph be defined as

$$G = (V, E) : \begin{cases} V = \{v_i : [nodeid, nodeType, attributes]\} \\ E = \{e_i : [source, dest, edgeType, attributes]\} \end{cases} \quad (1)$$

where V is the set of graph vertices and E is the set of graph edges. For purpose of database storage, vertices and edges are stored in relational database tables.

7.2 Graph Subsets

The manipulation tools are designed to extract subsets of the network based on graph-theoretic properties of the network such as graph distance (e.g. "Find all nodes at a graph distance of 2 or less from a given node") and graph density (e.g. "Find all nodes embedded in subgraphs with given density").

Graph subsets are defined by a union of two conditional operators - the vertex condition and the edge condition:

$$g_{CV,CE} = \begin{cases} v_{CV} & v_i \in V, CV(v_i) = true \\ e_{CE} & e_i \in E, CE(e_i) = true \end{cases} \quad (2)$$

where $g_{CV,CE}$ is the graph subset containing the results of the subset operation, CV and CE are logical operations (i.e. **WHERE** statements in SQL) that return true if the given node or edge, respectively, are to be included in the subset.

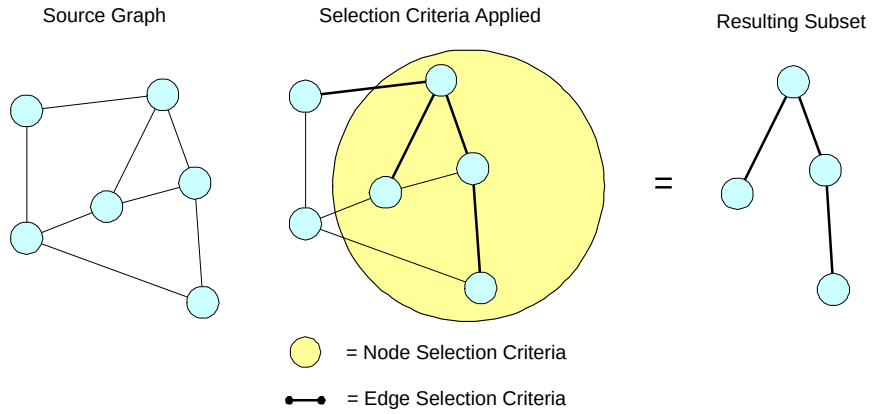

Figure 3: Graph Subset Creation

The CV and CE operations may place an arbitrary set of constraints on the selection of nodes and edges.

If $CV == null$ but $CE! = null$, a set of edges is extracted into the graph subset, and the vertex set consists of all vertices that are referred to as a source or destination of the selected edges.

If $CE == null$ but $CV! = null$, a set of vertices and all edges that connect them are extracted into the subset.

The graph subset as extracted by this operation is stored in a persistent database view and can be revisited or used as a basis for future pruning operations or queries.

7.3 EgoNet: Generating Ego Network subsets

The *EgoNet* operation returns a subset of the graph that consists of nodes that are located at or within a given graph distance. All edges that exist between these nodes are included in the subset, although it is possible to prune the subset further by applying an edge condition.

$$g_{egonet(v_c,d)} = \begin{cases} v_{egonet} & v_i \in V, distance(v_i, v_c) < d \\ e_{egonet} & e_i \in E, source(e_i) \in v_{egonet} \wedge dest(e_i) \in v_{egonet} \end{cases} \quad (3)$$

where v_c is the center node specified by the user, and d is the maximum graph distance between the center node and any other node in the subset.

It is a special case of the subset operations, as it imposes a relational condition upon the selection of nodes and requires computation of graph distance as an intrinsic database operation.

The *EgoNet* function performs a breadth-first search of the graph, starting at the center and expanding its search radius until it is equal to the specified maximum radius d.

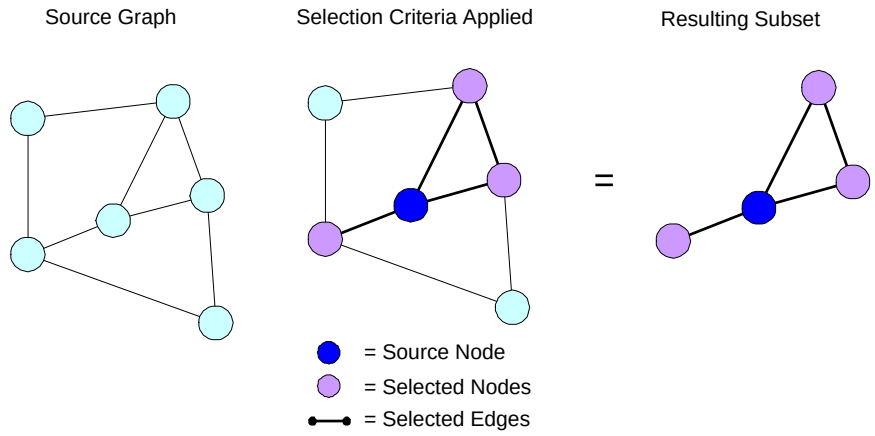

Figure 4: Extraction of EgoNet

Pure SQL is not very suitable graph-theoretic computations as most of them require recursion, which is expressly forbidden in SQL semantics. To implement graph traversals within the database, the recursive component is written in PL-SQL, a procedural language shared between Oracle and a number of other database engines. The PL-SQL function then calls SQL *SubSet* operations and builds up recursive views of the database.

Format:	`egonet(center-node-id,distance)`
Arguments: *center-node-id* ⇒ string *distance* ⇒ int	 the ID of the center node of the ego network maximum graph distance between the center node and any other node in the subset
Returns:	A **SELECT** set of objects of type **Node**.
Example Usage SQL:	 `SELECT * FROM egonet('bin_ladin',2);`
db2dynetml:	`db2dynetml <database> <output file> -ego bin_ladin -distance 2`
WWW:	In the node list, click on "Egonet" button next to the center node

7.4 Network Expansion

The purpose of *NetExpand* operation is, starting with an incompletely defined graph subset, find all surrounding network features. For example, *NetExpand* can be used to find all organizations, resources, locations and tasks connected to a given group of people.

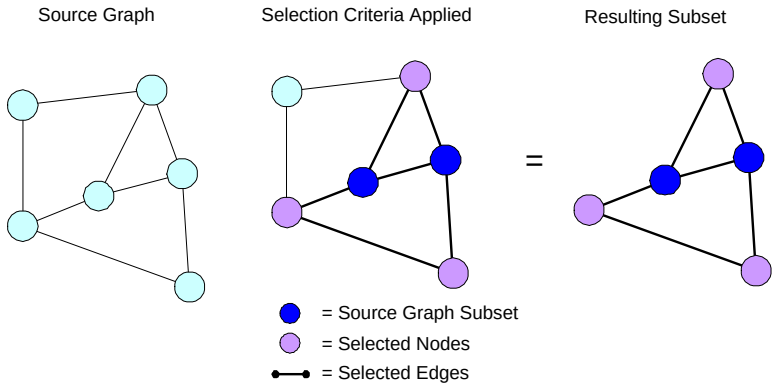

Figure 5: Network Expansion

The *NetExpand* function takes a graph subset S_source returns a subset of the graph S_result such that all nodes in S_result are at or within a given graph distance from one of the nodes in S_source. In a simpler fashion, it can be defined as a union of ego networks (*EgoNet* of all nodes in the S_source:

$$NetExpand(S_{source}, d) = \bigcup (EgoNet(v_i, d), v_i \in S_{source} \quad (4)$$

Format:	`fullnet(source-nodeset,distance)`
Arguments:	
center-node-set ⇒ name	the name of database view of type **Node** that contains the center nodeset. Database view can be created by running the following command: `CREATE VIEW <name> AS SELECT * FROM NODE WHERE ...node conditional...`
distance ⇒ int	maximum graph distance between the center nodes and any other node in the subset
Returns:	A **SELECT** set of objects of type **Node**.
Example Usage	
SQL:	`SELECT * FROM fullnet(source-nodeset,2);`
db2dynetml:	`db2dynetml <database> <output file> -net source.xml -distance 2`
WWW:	Load a database subset on screen by importing a DyNetML file, running queries or manual selection; then click on "Expand Network" in the toolbar

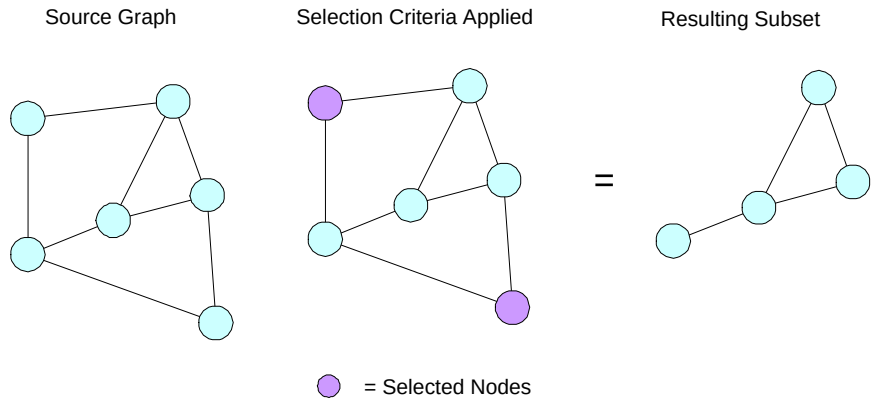

Figure 6: Pruning the network subsets

7.5 Nodeset Pruning

The *ExcludeCond* function removes nodes from a graph subset given a based on a selection criteria. The function is defined as:

$$g_{ExcludeCond}(source, CV) = \\ = \begin{cases} v_{exclude} & v_i \in source, CV(v_i) = false \\ e_{exclude} & e_i \in E, source(e_i) \in v_{exclude} \wedge dest(e_i) \in v_{exclude} \end{cases} \quad (5)$$

Format:	**exclude-cond(source-nodeset,exclude-conditional)**
Arguments: *source-nodeset* → name	the name of database view of type **Node** that contains the center nodeset.
exclude-conditional → sql	SQL statement containing the exclusion condition
Returns:	A **SELECT** set of objects of type **Node**.
Example Usage SQL:	**SELECT * FROM exclude-cond(source-nodeset,"WHERE nodeID!='bin_ladin'");**
db2dynetml:	**db2dynetml database-name output-file -net source.xml -exclude-cond "WHERE nodeID!='bin_ladin'"**
WWW:	Currently no Web interface implemented

The *ExcludeNodeset* function removes nodes from a graph subset given a based on membership in another set. The function is defined as:

$$g_{ExcludeNodeset}(source, exclude-list) = \\ = \begin{cases} v_{exclude} & v_i \in source, v_i \notin exclude-list \\ e_{exclude} & e_i \in E, source(e_i) \in v_{exclude} \wedge dest(e_i) \in v_{exclude} \end{cases} \quad (6)$$

Format:	**exclude-nodeset(source-nodeset,exclude-nodeset)**
Arguments:	
source-nodeset → name	the name of database view of type **Node** that contains the center nodeset.
exclude-nodeset → name	the name of a database view containing nodes to exclude from the given nodeset
Returns:	A **SELECT** set of objects of type **Node**.
Example Usage	
SQL:	**SELECT * FROM exclude-cond(source-nodeset,exclude-nodeset);**
db2dynetml:	**db2dynetml database-name output-file -net source.xml -exclude-list exclude.xml**
WWW:	Load the source nodeset from a DyNetML file, click on "Limit by" button in the toolbar and select a DyNetML file containing the exclusion list.

8 Auxiliary Tools

Current subsetting tools are written as stored procedures, and can be accessed through any programmatic interface to PosgreSQL, the Web interface or through a command-line programme **db2dynetml**.

The manipulation tools allow easy completion of incomplete datasets (e.g. "Given a set of people, find all organizations and resources connected to them"). To initiate dataset completion, the **db2dynetml** tool is launched with a DyNetML file containing the subject dataset. It then runs a set of graph-traversal expansions on the network and stores their results as a new network.

8.1 Exporting data from the database: db2dynetml

db2dynetml is a cross-platform command-line tool that exports information stored in a NetIntel database into a DyNetML file. Versions of the tool exist for Windows NT/2000/XP, Linux and OpenBSD; other operating systems that utilize a standard GNU compiler architecture can be also supported.

8.1.1 Command Usage

```
prompt> db2dynetml database-name output-file
 [-h<db host> ] [-u<db username> ] [-ego<egonet center> ]
 [-distance<egonet distance> ] [-doc<documentID> ]
 [- net<DyNetML file>] [-exclude<Exclusion List>]
```

The command-line parameters are explained below:

`database-name`	name of the database to connect to; **Required**
`-h db_host`	name or IP address of the database host (Optional; default value is the the name of the central CASOS application server)
`-u db_username`	database username (Optional; specify when not using a default host. The program will request a password interactively if the database requires password authentication
`output_file`	name of a DyNetML output file; **Required**
`-doc documentID`	Extract a dataset that is related to a particular source document
`-ego egonet_center`	Generate an ego network centered around a node (see section 7.3
`-distance egonet_distance`	Radius of the ego network; requires -ego or -net
`-net DyNetML_file`	expand a network from one specified in the file (see section 7.4
.	*Note: -ego and -net are mutually exclusive*
`-exclude Exclusion_List`	prune dataset (resulting from -net, -ego or -doc options) using exclusion list stored in a DyNetML file.

8.2 Importing Data into the database

dynetml2db is a tool for importing data stored in DyNetML files into the database. The database will not only store the data contained in the DyNetML file but also track the origin of data by creating an entry in the document table. Thus, the exact copy of the imported document can be retrieved using the `-doc` switch of the `db2dynetml`.

8.2.1 Command Usage

```
prompt> dynetml2db <database name> <input file>  [-h<db host> ]
```

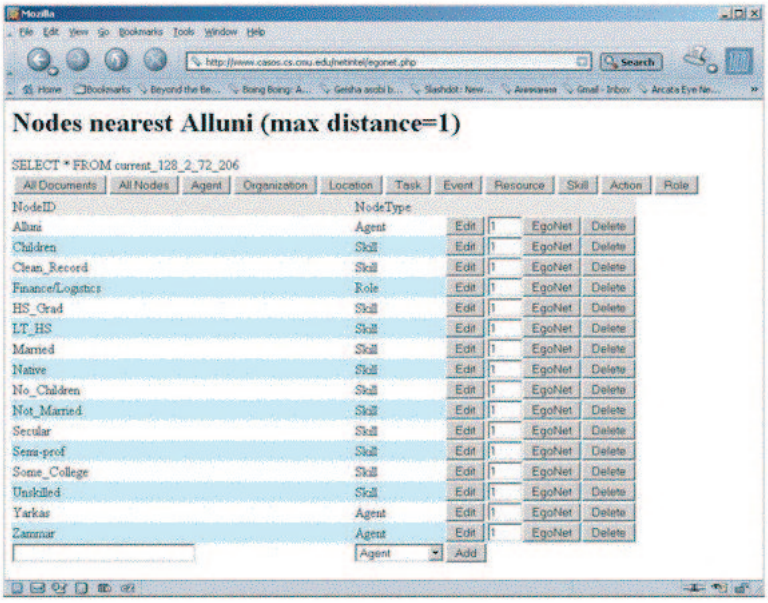

Figure 7: Screenshot of the WWW Interface to NetIntel Database

`[-u<db username> ] [-m<message> ]`

The command-line parameters are explained below:

`database-name`	name of the database to connect to; **Required**
`-h db_host`	name or IP address of the database host (Optional; default value is the the name of the central CASOS application server)
`-u db_username`	database username (Optional; specify when not using a default host. The program will request a password interactively if the database requires password authentication
`input_file`	name of a DyNetML file to be imported; **Required**
`-m message`	tag the imported data with a message; messages can be viewed and searched via the WWW interface

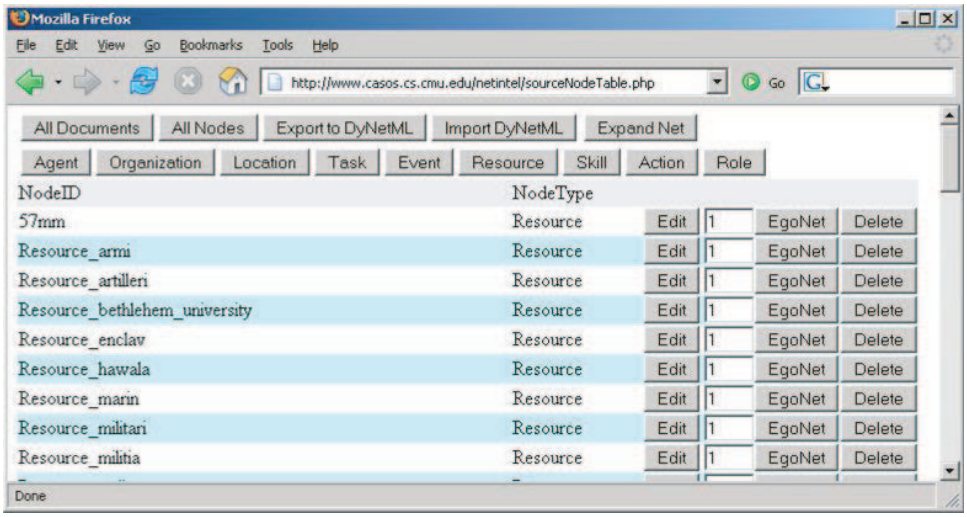

Figure 8: Screenshot: Main toolbar and Listing of Nodes

9 WWW Interface to NetIntel dataset

Figure 7 shows the WWW interface for entry, editing and manipulation of data stored in the NetIntel database. The interface allows a user to enter new nodes and edges, search the database for occurrence of keywords and build subsets of data based on attribute values as well as graph-theoretic measures.

The WWW interface allows graphical controls for building complex queries against the database, as well as easy import and export of data.

The WWW interface is written as in PHP and runs on an Apache server.

9.1 Managing Graph Data

The main screen (Figure 8) of the NetIntel interface consists of a toolbar that contains a number of common actions, and a list of graph nodes or objects in the database.

The toolbar of the NetIntel interface allows the user to import and export data, as well as run queries and run graph expansion:

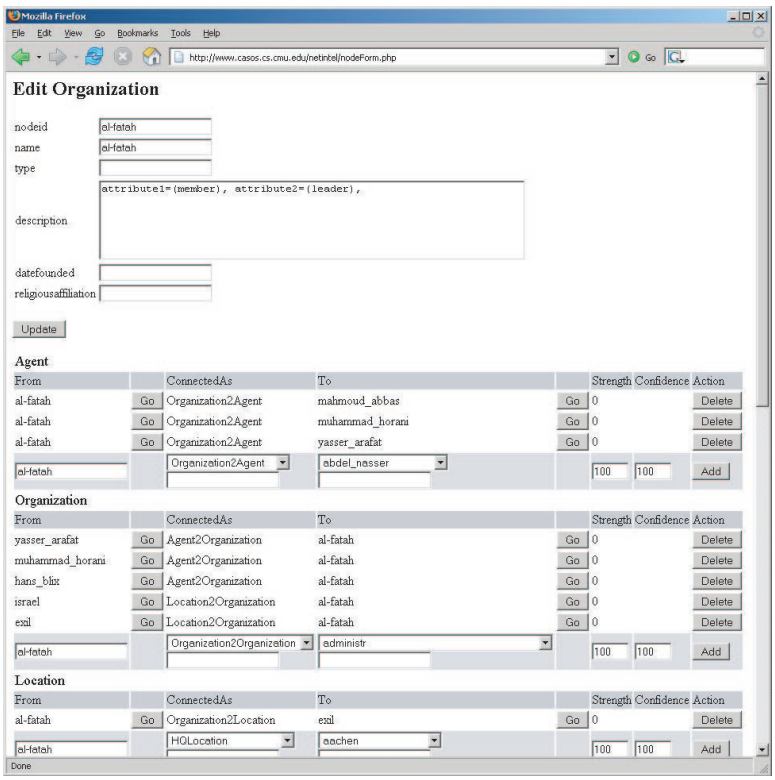

Figure 9: Screenshot: Viewing and editing information in an imported document

`All Documents`	Displays the list of documents imported into the database (see section 9.2)
`All Nodes`	Reset the current subset and display all of the nodes in the database
`Export to DyNetML`	Construct a DyNetML file from the currently selected subset of the network
`Import DyNetML`	Import a DyNetML file and register it as a data source document
`Expand Net`	Run the NetExpand operation (see section 7.4)
`Agent, Organization, Location, etc`	Limit the nodes displayed by node type. A button will be created for every type of nodes present in the database

The node list contains nodes in the currently selected database subset. The current subset can be manipulated using any of the query, network expansion and sorting functions. The current subset can also be exported as a DyNetML file using the `Export to DyNetML` button on the toolbar

Each of the nodes in the list is displayed with its ID, type and 3 buttons that allow operations on individual nodes:

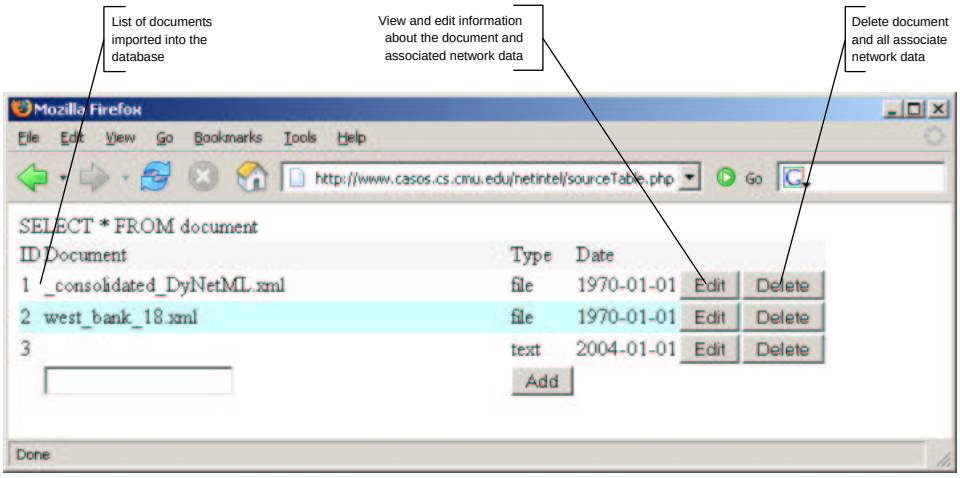

Figure 10: Screenshot: List of imported documents

`Edit`	Brings up the editing screen (see below)
`EgoNet`	Runs the EgoNet function (section 7.3) on the selected node and makes EgoNet selection the current subset. The distance parameter of the EgoNet function can be entered to the left of the button
`Delete`	Deletes a node and all edges associated with it.

The node editing screen (figure 9) allows the user to edit the attributes of a node and view and edit edges associated with the node. Number and type of attributes are dictated by the type of the node.

The list of edges is displayed below the node attributes. The edges are sorted by the type of target nodes (e.g. links between an agent and organization will be shown in the Organization section). Each of the edges is listed with its label, as well as strength and confidence values. Pressing the `GO` button next to the edge target displays the target node in the editing screen.

This behavior of the interface allows for fast manual data entry from text sources. For example, to code a statement *"Hamas was headed by Yassin and Rantissi"*, the user should add a node of type **Organization** and name it *Hamas*. The new node will be automatically displayed in the editing screen. In the **Agent** category of the edge list, the user should type the name *Yassin* and add the edge. Then, *Rantissi* is added with the similar operation. Both nodes will be automatically added to the database and can be edited by pressing the `GO` button next to them.

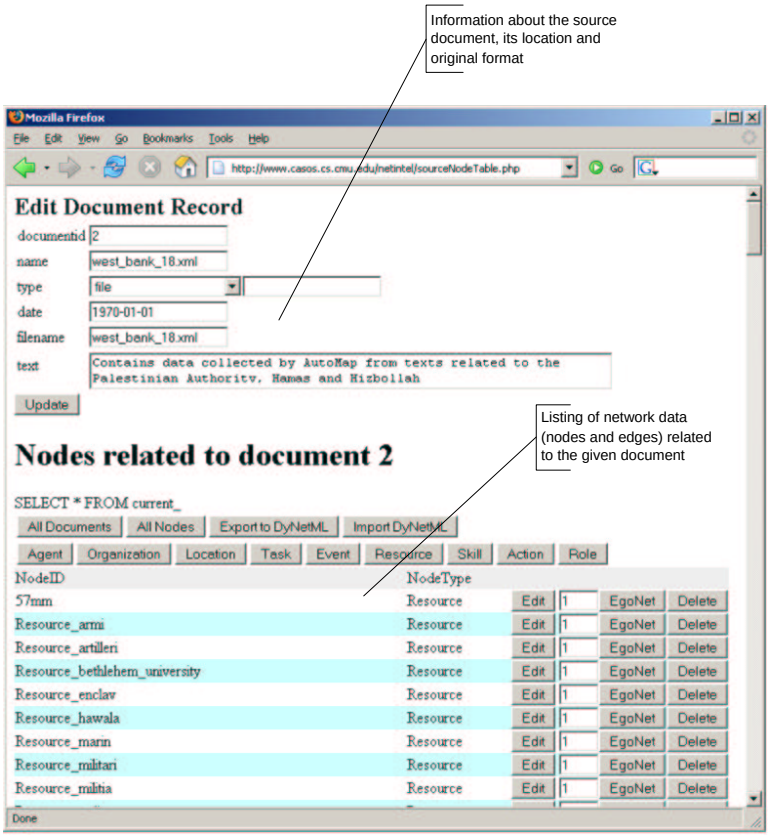

Figure 11: Screenshot: Viewing and editing information in an imported document

9.2 Managing Data Source Documents

NetIntel interface allows the user to view and manage any number of data source documents after they have been imported into the database. This makes it possible to integrate multiple data sources as well as retain access to each dataset individually. Figures 10 and 11 show the interface for managing collections of documents and associated network data.

10 Conclusion

NetIntel is a flexible database system designed for handling large volumes of graph-based and social network data. NetIntel is built as an extension to a standard SQL database, and thus does not require custom or experimental database storage engines, and can utilize the wealth of third-party interface software and APIs. While not as full-featured as QGraph or other systems designed specif-

ically for graph manipulation, it provides a reliable means of storage and manipulation of graph-based data and easy-to-use facilities for export of data into analysis tools as well as online browsing and data entry.

References

[1] Postgresql. www.postgresql.org.

[2] Serge Abiteboul, Dallan Quass, Jason McHugh, Jennifer Widom, and Janet L. Wiener. The Lorel query language for semistructured data. *International Journal on Digital Libraries*, 1(1):68–88, 1997.

[3] J. Diesner K.M. Carley. Automap1.2 - extract, analyze, represent, and compare mental models from texts. Technical Report CMU-ISRI-04-100, Carnegie Mellon University, School of Computer Science, Institute for Software Research International, 2004.

[4] J. Diesner K.M. Carley. *Revealing Social Structure from Texts: Meta-Matrix Text Analysis as a novel method for Network Text Analysis.*, chapter 4. Causal Mapping for Information Systems and Technology Research: Approaches, Advances, and Illustrations, V.K Naraynan, D.J. Armstrong (Eds.). Idea Group Publishing, Harrisburg, PA, 2005.

[5] Kathleen M. Carley. Smart agents and organizations of the future. In Leah Lievrouw and Sonia Livingstone, editors, *The Handbook of New Media*, chapter 12, pages 206–220. Sage, Thousand Oaks, CA, 2002.

[6] I2 Corporation. Analyst's notebook: Powering your analysis. http://www.i2inc.com/Products/Analysts_Notebook/default.asp.

[7] IntelCenter Corporation. Intelcenter database (icd). http://www.intelcenter.com/icd/index.html.

[8] RAND Corporation. Purpose and description of information found in the incident databases. http://www.tkb.org/RandSummary.jsp.

[9] Kathleen M. Carley Craig Schreiber. Construct - a multi-agent network model for the co-evolution of agents and socio-cultural environments. Technical Report CMU-ISRI-04-109, Carnegie Mellon University, School of Computer Science, Institute for Software Research International, Pittsburgh, PA, 2004.

[10] Le Gruenwald, Gary McNutt, and Adrien Mercier. Using an ontology to improve search in a terrorism database system. *Proceedings of the 14th International Workshop on Database and Expert System Applications (DEXA'03)*, 2003.

[11] Brian Houghton. Understanding the terrorism database. National Memorial Institute for Prevention of Terrorism Quarterly Bulletin, 2002.

[12] IntelCenter.com. Mapping al-qaeda v1.0. www.intelcenter.com.

[13] Serge Abiteboul Jennifer Widom. Lore: a database management system (dbms) for xml. http://www-db.stanford.edu/lore/.

[14] Blau Immerman Jensen. A visual language for querying and updating graphs.

[15] K.M. Carley M. Tsvetovat. Modeling complex socio-technical systems using multi-agent simulation methods. *Kunstliche Intelligenz (Artificial Intelligence Journal)*, Special Issue on Applications of Intelligent Agents(2), 2004.

[16] Jim Melton. Sql language summary. *ACM Computing Surveys*, 28(1), March 1996.

[17] "J. Neville and D. Jensen". Supporting relational knowledge discovery: Lessons in architecture and algorithm design. *Papers of the ICML 2002 Workshop on Data Mining Lessons Learned*, 2002.

[18] Brent L. Smith and Kelly R. Damphousse. The american terrorism study: Indictment database, 2002.