Database Support for Exploring Criminal Networks

M. N. Smith, P. J. H. King

School of Computer Science & Information Systems,
Birkbeck College, University of London
{mat, pjhk}@dcs.bbk.ac.uk
http://www.dcs.bbk.ac.uk

Abstract. Understanding the data gathered thus far in a criminal investigation is of great importance, particularly in terms of guiding its future course. One established method of comprehending such data is to incrementally visualize it as a network to ascertain if, and how, objects of interest are connected. In this paper we describe this form of visual data analysis, discuss why current database support for the method is inadequate, present an experimental database query system for exploring data in this manner, and outline areas of future research and development.

1 Introduction

Data is often best comprehended when presented as a network. This is a consequence of the connections between the objects of interest, such as people or locations, being of as great an importance, if not more so, as the information known about the objects, such as the age or occupation of a person. For example, if we have the information that David is a friend of Andrea, who is the sister of Paul, who frequents the King George public house, as does David, Fiona and Edward, who works at Daltons bank where he has a colleague called Christine, who is married to Brian, who is the brother of Angela, who is Fiona's friend, then this information is best presented as a network such as that displayed in Fig. 1.

Presenting such data in this manner increases comprehensibility, providing that the data set is not too large, and enables the reader to establish visually if objects are connected. Thus, if we ask if there is a connection between Christine and Paul who we believe are not known to each other, the possible connections are easily seen.

Visualizing a social network [1] in this way is of importance in the analysis of data relating to criminal activities. The fundamental approach used, which is now used in other fields such as forensic accounting [2], is to incrementally construct a meaningful subset of the data gathered thus far using a diagram similar to that displayed in Fig. 1, which is referred to as a *link chart*, in order to better understand the connections between the objects of interest and to suggest productive lines of further enquiry.

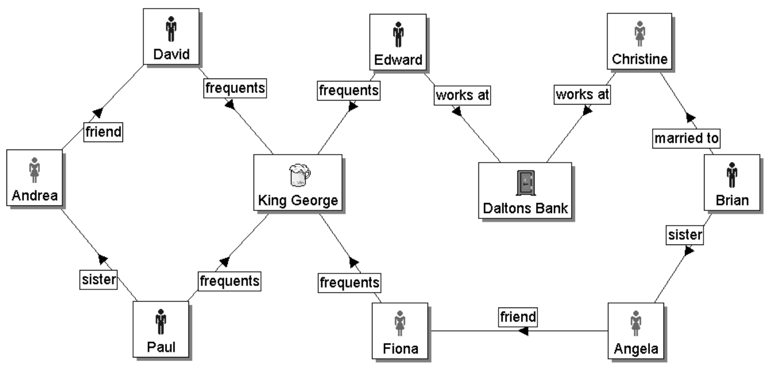


Fig. 1. The connections between a set of people displayed as a network

Link charts are typically constructed incrementally. The objects and links that are included in a chart are often altered during the lifetime of the chart, both as part of the natural exploratory nature of the knowledge discovery process and to reflect the acquisition of new information. Thus, if the underlying data is stored in a database one can consider a link chart to represent the results, or a subset of the results, returned by a number of queries such as *display all those paths that connect David and Fiona*, and *display all those people who are directly linked to Brian*, each of which may add one or more objects or links to the chart.

2 Representing Networks within Databases

The data stored in a database conforms to an abstract model of the problem domain referred to as the *database schema*. The manner in which the problem domain is modeled in a database schema depends on the *data model* used. Binary-relational data models [3, 4, 5] are based upon two concepts, entity types and relationship types. Entity types are used to model categories of objects in the problem domain that are uniquely identifiable and have an independent existence, relationship types are used to model the relationships between these objects. For example, the following binary-relational database schema models the data displayed in Fig. 1.

```
entity type location
entity type person

relationship type   frequents     person -> location
relationship type   works at      person -> location
relationship type   married to    person -> person
relationship type   friend        person -> person
relationship type   sister        person -> person
```

A database using a binary-relational data model may be regarded as a network, where nodes are represented by instances of entity types and edges by instances of relationship types; all of which are uniquely identified. This is analogous to how data is displayed in a link chart. Hence, the design and maintenance of a system for visually analyzing data in the manner described in section 1 is greatly simplified by using such a database, as there is no need to convert between the representation of data used in the database and that used in the visualization system.

This is not the case with record-based data models [6], such as the relational model of data [7], network data models, and object-orientated data models, where there is a mismatch between the model of data in the database and that used in the visualization system. In order to manage this mismatch an additional software component has to be introduced to convert between the two representations. This increases maintenance costs, as the component has to be reconfigured whenever the structure of the database schema is modified. It can also adversely affect performance. Thus, while this approach may be used for small databases, it is not applicable when large amounts of data need to be analyzed, such as in telephone call analysis.

3 Database Query Languages

Database query languages such as SQL [8] assume the schema of the database queried is known and that a query can be expressed as a single statement. They are most suitable for queries such as *retrieve the names of all customers that placed orders of more than $10,000 in the last financial year*, or *on what date did James Smith join our company*, where specific pieces of information are required. The range of queries that may be expressed in such languages is often restricted to that which may be expressed in the relational algebra [7] extended with a fixpoint operator [9] and operators for grouping and generating new values so that the rapid processing requirements of these applications can be obtained.

Queries to retrieve paths of an undefined length and structure through a database, such as *display all those paths that connect David and Fiona,* cannot be expressed in query languages with only this expressive power. The majority of visual database query systems [10], in addition to providing inappropriate result display mechanisms for the style of data exploration we are discussing in this paper, only support visual variants of query languages such as SQL. Thus, they are not well suited for supporting the exploration of data such as that presented in Fig. 1.

4 Current Software Support for Link Chart Analysis

Several software products [11, 12] support the construction and analysis of link charts. These products provide a wide range of data visualization and analysis facilities that enable the incremental development of charts that can be stored, and retrieved for subsequent display and further development.

Facilities for retrieving the data displayed in a chart from a database using the relational model of data are also provided. However, the expressive power of these facilities is a subset of the query languages described in section 3, and as discussed in section 2 an additional software component is provided which has to be configured to the structure of the particular database being accessed; hence, limiting the scalability of the products. Whilst facilities to ascertain if two objects are connected are provided they do not consider the data stored in the database but only that displayed in the chart. This is useful for highlighting paths in large charts but otherwise of limited use. Thus, the analysis facilities provided in these products are largely chart focused.

This is in contrast to the research we report in this paper, which is database focussed. In our approach [13] the data analysis takes place in the database, and not just within a link chart. The chart is the mechanism used to display the results of the analysis and to integrate these results with other information retrieved or supplied directly.

5 The Exploratory Database View Constructor

The Exploratory Database View Constructor (EDVC) is an experimental database query system designed to support the incremental style of visual data analysis described in section 1. The following subsections describe the data model used in the system, the two distinct styles of query interface provided, and the implementation.

5.1 The Data Model of EDVC

The problem domain is modeled as a set of object types and link types, which are equivalent to entity types and relationship types in a binary-relational data model. Objects types model categories of objects in the problem domain that have an independent existence, are uniquely identifiable, and would normally have information stored about them. An object type may be declared a *subtype* of one or more other object types, which are referred to as the *supertypes* of the object type. An object that is an instance of the subtype is also regarded as an instance of the supertypes. For example, if the object type *police officer* is declared a subtype of the object type *person* then all objects that are instances of object type *police officer* are also regarded as instances of the object type *person*.

Link types model directed relationships between objects in the problem domain, and may be defined from either an object type or an object to either an object type or a value type, which is a system-defined data type such as text or number. Link types defined from an object are used to model relationships that may only be appropriate for that object. For example, the following database schema models personal relationships, and allows the age and occupation of each person to be stored as well as the maiden name of particular person Christine:

```
object type  person

link type    married to   person -> person
link type    friend       person -> person
link type    sister       person -> person
link type    age          person -> number
link type    occupation   person -> text

link type    maiden name  Christine -> text
```

A link type may have one or more attributes defined, each of which has an associated name and value type, which model the information that may be stored directly about an instance of the relationship modeled. For example, in the database schema given above we could have defined an attribute for the link type friend to indicate the strength of a friendship.

5.2 The Explorer Query Interface

The explorer query interface provides a browsing style of interaction that allows a link chart to be incrementally constructed, possibly over several sessions as a link chart may be saved to file. Several explorer interfaces can be open at the same time, each displaying a link chart that may reflect a different view of the data stored.

The user may begin by adding one or more objects to an empty chart or by displaying the objects linked to a number of other objects; the number of links required is specified using a condition such as greater than three, and the type of links considered may be restricted. Objects displayed with a question mark enclosed within a circle, such as those in Fig. 2, are involved in one or more links stored in the database that are not displayed on the chart.

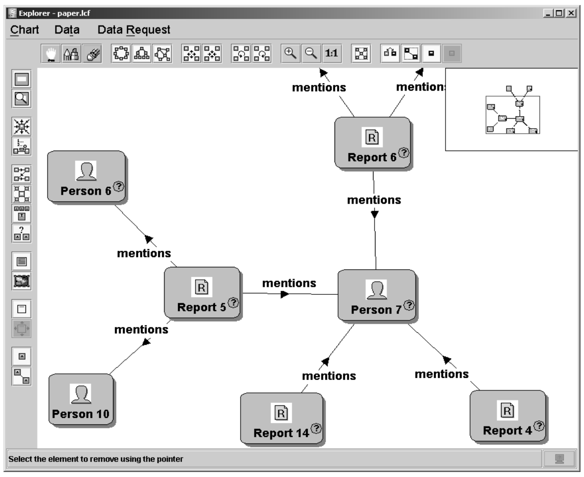


Fig. 2. The explorer query interface

The connections such objects have to other objects can be explored using the facilities described in the following subsections.

5.2.1 Display the Information Stored Directly about an Object

The objects and values that are directly linked to a selected object may be displayed in a separate dialog, and if required than added to the chart. For example, Fig. 3 displays the chart given in Fig. 2 with additional information regarding Person 7 added.

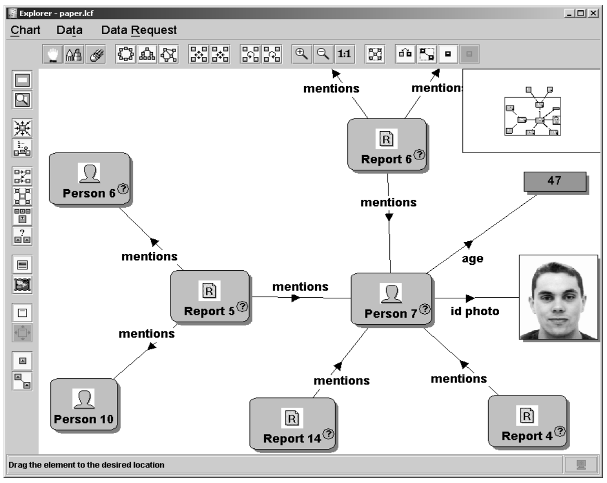


Fig. 3. The chart displayed in Fig. 2 with additional information regarding Person 7 added

5.2.2 Display the Objects Connected to a Selected Object

The objects, and the connecting paths, that are directly or indirectly connected to a selected object may be added to the chart. The type and direction of the links in a connecting path may be constrained, as may the maximum length of a connecting path. For example, displaying the people connected to a suspect in an investigation could prove useful in understanding the social context of the suspect.

5.2.3 Display the Paths Connecting Two Selected Objects

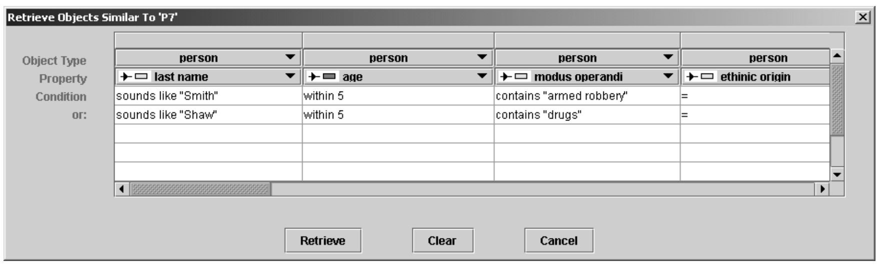
The paths connecting two selected objects may be added to the chart. The type and direction of the links in a connecting path may be constrained, as may the maximum length of a connecting path. For example, it is often useful to ascertain if, and how, two suspects in an investigation are connected.

5.2.4 Display the Objects Connected to Two or More Selected Objects

The objects, and the connecting paths, that are connected to two or more selected objects may be added to the chart. The type and direction of the links in a connecting path may be constrained, as may the maximum length of a connecting path. For example, if an armed bank robbery involving six men had occurred and four of the men had been identified it would be helpful to display the people closely connected to these four men so that a list of possible suspects for the other two could be created.

5.2.5 Display the Objects Similar to a Selected Object

The objects deemed similar to a selected object according to a given set of similarity conditions relating to the information stored directly about an object may be displayed in a separate dialog, and then if required added to the chart. For example, identifying people who have similar characteristics to that of a known criminal, or a criminal profile, may prove productive in identifying previously unknown identities for a criminal, or for creating a list of possible suspects. Similarity conditions are combined using a condition grid similar to that in Fig. 4.


Retrieve Objects Similar To 'P7'

Object Type	person	person	person	person
Property	last name	age	modus operandi	ethinic origin
Condition	sounds like "Smith"	within 5	contains "armed robbery"	=
or:	sounds like "Shaw"	within 5	contains "drugs"	=

Fig. 4. A condition grid specifying similarity conditions

Only objects that satisfy all the conditions specified in at least one of the rows in the grid are displayed. Boolean connectives are not explicitly used as their semantics when used in natural language differs from that when used in a database query language, which can lead to errors [14].

5.2.6 Display the Common Properties of Two Selected Objects

The objects and values linked to two selected objects by the same type of link may be displayed in a separate dialog, and then if required added to the chart. For example, this facility can be used to ascertain the level of similarity between two people.

5.2.7 Editing a Link Chart

Although the query facilities allow the objects and links displayed to be constrained not all of those displayed will necessarily be considered relevant by the user. Therefore, any element on a link chart may be removed, without affecting other charts or the data stored. The clarity of a chart can also be improved be altering the position of elements, either by dragging them to better positions or by using one of the automatic chart layout facilities provided which may rearrange the elements in a circular fashion, hierarchically, or to minimize the number of links that cross.

The comprehensibility of a chart may be enhanced by adding elements other than those representing the objects and links stored in the database. These include text notes to supply information in the form of free text, labeled boxes to indicate the grouping of objects, and the addition of objects and links not stored to see what impact they would have on the network being explored.

5.2.8 Extending the Range of Query Facilities Supported

The range of query facilities supported in the query interface may be extended. A programming framework is provided that allows third party software components, implemented using the Java™ programming language, to retrieve objects and links from the database and display them in a link chart. Search parameters may be specified by selecting elements in a link chart and by entering values via a dialog.

5.2.9 The Execution of Queries

Each query in the explorer query interface runs in a separate thread of execution. This allows the user to explore the data stored while one or more queries are executing, and for the results returned thus far by a query to be displayed. In practice this would not generally be needed for the query facilities described above as they run in polynomial time with respect to the size of the database[1] and the length of a connecting path would typically be constrained. However, if software components supporting computationally complex operations were provided, such as components for finding *cliques* [16], executing each query in a separate thread of execution would prove useful.

5.3 Filter Patterns

The range of queries that may be expressed in the explorer query interface is restricted to those that are supported by the query facilities provided and by third party software components. Therefore, a second style of query interface is provided that allows ad-hoc *filter patterns* to be constructed that may filter the objects and links stored in the database, or those displayed in a link chart, so that only those objects and links that match the pattern specified are displayed. A filter pattern consists of a set of constraints, of which there are several types.

5.3.1 Object Constraints

An object constraint may match a particular object, all objects that are instances of a given object type, or all objects, regardless of type. Object constraints that match all objects that are instances of a given object type may have one or more associated *filter conditions*, which refer to the information stored directly about the objects of the appropriate type and restrict the objects matched to those that match the conditions specified. Filter conditions are combined in a grid similar to that in Fig. 4.

[1] Non-trivial social networks tend to be sparse by nature [15]; thus the runtime of these operations would generally be quicker than polynomial time.

5.3.2 Link Constraints

Link constraints connect two object constraints and dictate that the objects matched by the associated object constraints must be directly linked. Link constraints may take the direction of the link into consideration, and can restrict the links matched to only those with a given label, or those that are instances of a given link type. For example, Fig. 5 displays a filter pattern for telephone call analysis.

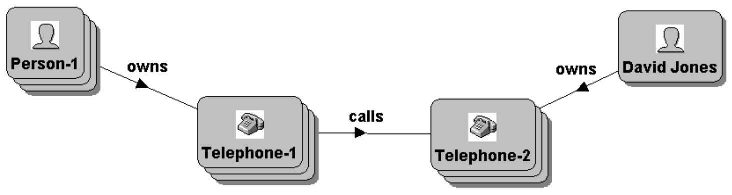


Fig. 5. A filter pattern that would display all people who telephoned David Jones, and the associated telephones and links. Object constraints that only match instances of a given object type are uniquely labeled with the name of the object type followed by a hyphen and an integer.

Link constraints that only match links that are instances of a given link type may have one or more filter conditions, which restrict the links of the appropriate type matched to only those whose attribute values that satisfy the conditions specified. Link constraint filter conditions are combined in a manner similar to those of object constraints.

5.3.3 Path Constraints

Path constraints are a more general form of link constraint. They connect two object constraints and dictate that the objects matched by the associated object constraints must be connected. The path connecting the objects may be of any length and structure, although they may be restricted to only those whose length is less than a given maximum and satisfy a given regular expression, which is expressed using a graphical notation similar to that used in formal language theory [17].

Path constraints are required for matching undefined connections between objects, which are difficult to express using only object and link constraints, particularly if the user is not familiar with the database schema, as the length and structure of a connecting path cannot typically be predicted in advance.

5.3.4 Additional Refinements

A number of additional refinements can be made to a filter pattern. One or more of the constraints in a filter pattern may be *excluded*, which dictates that the objects or links matched by the constraint are not displayed in the corresponding link chart. Also one or more object constraint comparison edges can be added, each of which links two object constraints and specifies that the objects matched by the constraints must either differ or be the same.

5.3.5 Matching Filter Patterns

Finding a path that connects two nodes in a network that satisfies a regular expression is an operation that may execute in non-polynomial time with respect to the size of the database. Several classes of regular expression can be evaluated in polynomial time [18], and in practical use the length of a valid connecting path would typically be constrained. Therefore, a useful class of queries can be expressed[2] that may run in acceptable time periods for large databases.

5.3.6 The Filter/Display Cycle

The objects and links displayed after matching a filter pattern may be filtered further by refining the original filter pattern or by specifying a new pattern and filtering those previously displayed. Alternatively, the objects and links in the chart displayed may be added to, or removed, using the facilities described in section 5.2. This permits link charts to be constructed that reflect a sequence of queries that could not be expressed using just one of the query interfaces.

5.4 The System Implementation

EDVC is implemented using version 1.3 of the Java™ programming language and may run on any platform supporting the standard edition of the Java™ 2 platform. The classes in the system are logically divided into those used to provide the user interface and those used to perform the query operations, and the data exchanged between the two is independent of the underlying data source. Thus, EDVC may be used as an interface to any data source, or collection of data sources, for which implementations of the query classes are provided. The current implementations are for the database management system Sentences [20], the performance of which [21] is the primary factor effecting the runtime of the query facilities provided in EDVC, which are based upon concepts from graph theory [22].

6 Future Research & Development

In this section we describe areas of ongoing research and development to extend and enhance the work described in this paper relevant to the law enforcement community.

[2] The class of queries that may be expressed using filter patterns does not fit well into classification schemes such as that proposed by Chandra [19] as not all of the operators in the relational algebra are supported in EDVC, which focuses upon graph traversal.

6.1 Support for Hypothesis Testing

The database management system currently used as the underlying data store represents each database as a *profile*, which consists of a set of *chapters*, each of which may store one or more objects and links, or information regarding those previously stored. This has great potential for use in the analysis of criminal networks.

Criminal networks are often dynamic with boundaries to the legitimate world that are hard to define [23]. By providing each investigator with a separate profile which contains one or more chapters storing the information generally available, and an additional chapter that stores the modifications made by the investigator, the impact that any modifications made have on the network analyzed can be ascertained without effecting the view that other users have of the common data.

6.2 Support for Personalized User Views

The relationships in the problem domain that are relevance to a particular user may correspond to two or more of the links stored. For example, consider the data displayed in Fig. 1. An investigator may only wish to see the possible associates of each person, which he or she may define as the people who frequent the same public house or work at the same location. Facilities for defining new types of links generated from two or more of those stored would reduce the number of objects and links displayed, increasing comprehensibility while still retaining the essential information.

7 Concluding Remarks

The motivation behind the work presented in this paper was to provide better database support for an established method of visually analysing data that is typically stored in a database. The incremental style of data exploration supported may, and has been, used in fields other than that of the analysis of crime related data. Thus, EDVC may be considered a general-purpose database query system, and not one specifically tailored to the analysis of crime related data. We are currently undertaking a detailed user evaluation of EDVC to test the usability of the facilities provided.

Acknowledgements

The authors would like to thank IBM and the Engineering and Physical Sciences Research Council for financial support, and Lazy Software Limited and Xanalys Limited for the use of their software products for research and evaluation.

References

1. L. C. Freeman: Visualizing Social Networks, Journal of Social Structure Vol. 1 No. 1, 2000.
2. Forensic Accounting: http://www.forensicaccounting.com.
3. J. R. Abrial: Data Semantics, Proceedings of the IFIP Working Conference on DBMS, 1974.
4. D. R. McGregor, R. G. Malone: The FACT Database System, Proceedings of Symposium for Database Systems, Systems for Large Databases, North Holland, 1980.
5. S. Williams: The Associative Model of Data, Lazy Software, ISBN 1903453003, 2000.
6. W. Kent: Limitations of Record-Based Information Models, ACM Transactions on Database Systems Vol. 4 No. 1, 1979.
7. E. F. Codd: A Relational Model of Data for Large Shared Data Banks, Communications of the ACM Vol. 13 No. 6, 1970.
8. C. J. Date, H. Darwen: A Guide to the SQL Standard, Addison Wesley, ISBN 0201964260, 1997.
9. V. Aho, J. D. Ullman: Universality of Data Retrieval Languages, Proceedings of the 6th ACM Symposium on Principles of Programming Languages, 1979.
10. T. Catarci, M. F. Constabile, S. Levialdi, C. Batini: Visual Query Systems for Databases: A Survey, Journal of Visual Languages and Computing Vol. 8 No. 2, 1997.
11. Watson: Xanalys Limited, http://www.xanalys.com.
12. Analyst's Notebook: I2 Limited, http://www.i2group.com.
13. M. .N. Smith, P .J .H. King: The Exploratory Construction Of Database Views, Research Report BBKCS-02-02, School of Computer Science and Information Systems, Birkbeck College, University of London, available from http://www.dcs.bbk.ac.uk/TriStarp under publications, 2002.
14. S. Greene, S. Devlin, P. Cannata, L. Gomez: No IFs, ANDs, or ORs: A Study of Database Querying, International Journal of Man-Machine Studies Vol. 32 No. 3, 1990.
15. V. Batagelj, A. Mrvar: Pajek – Program for Large Network Analysis, Connections Vol. 21 No. 2, 1998.
16. R. Hanneman: Introduction to Social Network Analysis, http://wizard.ucr.edu/~rhannema/networks/nettext.pdf.
17. V. J. Rayward-Smith: A First Course in Formal Language Theory, Blackwell Scientific Publications, ISBN 0632011769, 1983.
18. A. O. Mendelzon, P. T. Wood: Finding Regular Simple Paths in Graph Databases, Proceedings of the Fifteenth International Conference on Very Large Data Bases, 1989.
19. A. K. Chandra: Theory of Database Queries, Proceedings of the 7th ACM Symposium on Principles of Database Systems, 1988.
20. Sentences: Lazy Software Limited, http://www.lazysoft.com.
21. Lazy Software Limited: Sizing Associative Databases, white paper available from http://www.lazysoft.com, 2002.
22. R. Diestel: Graph Theory, Springer-Verlag, ISBN 0387950141, 2000.
23. M. K. Sparrow: The Application of Network Analysis to Criminal Intelligence: an Assessment of the Prospects, Social Networks Vol. 13, 1991.