

USER MANUAL
VERSION 2009-05-25



Commetrix Data Model **Exploration of Network Dynamics and Contents**

Product Website: www.commetrix.net
Vendor Contact : info@trilexis.com



Contents

1	Introduction	3
2	Installation	3
2.1	MySQL Community Server.....	3
2.2	MySQL J/Connector.....	3
3	Commetrix Data Model	4
3.1	General Introduction	4
3.2	Table 'Network'	5
3.3	Table 'Node'	6
3.4	Table 'Linkevent'	6
3.5	Tables 'NetworkDetailConfig', 'NodeDetailConfig' and 'LinkeventDetailConfig'	7
3.6	Tables 'Keyword', 'SubjectKeyword' and 'ContentKeyword'	9
3.7	Table 'Stopword'	10
3.8	Tables 'Node_tmp', 'Linkevent_tmp', 'LinkeventSender_tmp', 'LinkeventRecipient_tmp' and 'LinkeventParent_tmp'	10
4	Creating Networks from a Commetrix database.....	10
4.1	Thread-based communication with only one node as sender	10
4.2	Thread-based communication with several nodes as senders	12
4.3	Sender-receiver communication with only one node as sender	14
4.4	Sender-receiver communication with several nodes as senders	16
4.5	Using linkevent-based communication for a co-authorship network.....	18
4.6	Using sender-receiver and thread-based communication for a citation network.....	20
5	Contact	24

1 Introduction

COMMETRIX is a software framework for the analysis of social networks. Next to conventional static network analysis of cumulative snapshots of relationship networks, its event-based network model enables the examination of network evolution over time.

Commetrix offers an innovative **DATA MODEL**. Unlike in most other SNA tools, it does not store links directly as valued relationships but as individual relational events. This provides the opportunity of modelling networking processes with multiple types of relationships and comprehensive qualitative and quantitative node and link attributes in a single dataset. With this approach, users can encode aspects like topic descriptors (e.g. content coding, keywords), types of links, e.g. socialization, document exchange, affiliation, media, time stamps of links, links connecting multiple nodes, as well as arbitrary quantitative or qualitative variables classifying actors, e.g. affiliation, age, types, etc.

The **USER INTERFACE** is very easy to use. It is developed to visually support exploratory examination of a network dataset in order to identify and observe relevant substructures, periods, and processes of your network data. Commetrix computes time window measures and additionally provides very sophisticated functionality for displaying and animating the community evolution as an evolving graph to visually inspect the actors' activities. The animated graph, called *communigraph*, is one of the best existing visualizations of network change. Visual variables can be set by the user to represent node and link properties by label, node size, node colour (brightness, transparency), or a number of rings around the node.

Complex options for time, actor, relationship, and topic filtering help to **FOCUS RELEVANT STRUCTURES**, i.e. relevant actors, relationships, time periods, or even topics. For any time period and for any selection, typical social network measures can be computed and analyzed. Selections can be exported to tables for further analysis. This includes the export of network changes over time. All these features help to actually represent and visually trace change in a network and adds additional insight to the quantitative results.

This manual gives an introduction to the Commetrix data model and how to store network data in a Commetrix database that can be visualized by the Commetrix Analyzer.

2 Installation of Commetrix Databases

2.1 MySQL Community Server

To store Commetrix network data and to create CMX files from them a MySQL database is required.

Download the MySQL Community Server (<http://dev.mysql.com/downloads/>) and follow the installation instructions given there. CMXDBProducer requires at least MySQL 5.0. We do not guarantee the CMXDBProducer working properly with older versions. Keep the adminname and password combination for the MySQL database as it is required when running the CMXDBProducer.

To monitor the network data in the database some GUI tools like the MySQL Query Browser might be helpful. They can be downloaded at <http://dev.mysql.com/downloads/gui-tools/5.0.html>.

2.2 MySQL J/Connector

To allow a Java-based program like the CMXDBProducer to access a MySQL database a JDBC driver has to be installed. Download the MySQL Connector/J (<http://dev.mysql.com/downloads/connector/j/>) and

follow the installation instructions given there. The connector will be a .jar file which has to be copied to <path-to-jre>/lib/ext folder of your Java Virtual Machine. CMXProducer requires J/Connector 3.0 to ensure all database functionalities. We do not guarantee the CMXProducer working properly with other versions.

3 Commetrix Data Model

In this section the Commetrix data model is explained. After referencing the example scripts that come with the software distribution, a general introduction to the basic concepts of nodes in a network being related by linkevents is given. Afterwards the different database tables and how to fill them with data from an external source are presented in more detail.

Example Scripts. An empty Commetrix database can be created by using the MySQL script 'CMX_SQLScript_create_empty_cmxdatabase.sql'. The data types and default values for each column can be found there. An example database which illustrates sender-receiver communication via email can be created using the MySQL scripts 'CMX_DBEXAMPLE.sql'.

3.1 General Introduction

Basically, a Commetrix network consists of the three data types network (table 'Network'), node (table 'Node') and linkevent (table 'Linkevent'). Nodes are related (linked) by linkevents, which are aggregated to links by the CMXDBProducer. Nodes and links are visualized as a graph by the CMXAnalyzer. A database can consist of more than one network, thus each table links via column 'NetworkID' to the appropriate entry in table 'Network'. However, having large datasets we recommend to use a separate database for each network.

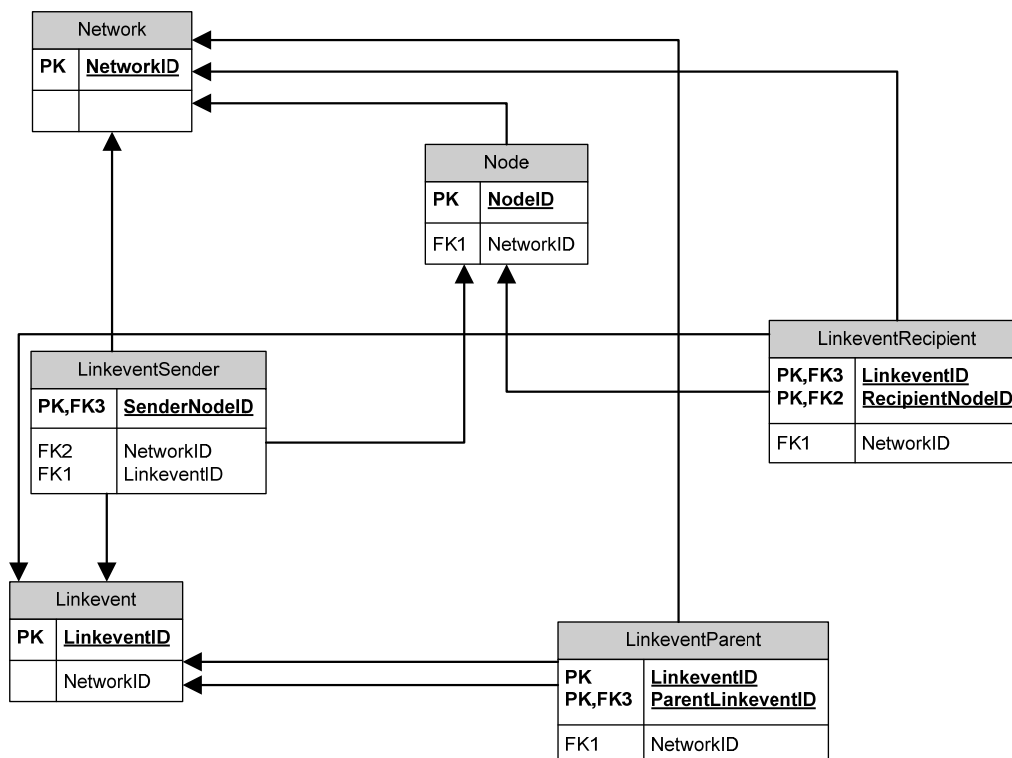


Figure 1: Commetrix data model – Basic data types and their relationships.

A linkevent can be originated by one or more nodes. This relation is stored in table 'LinkeventSender' where column 'LinkeventID' relates to the linkevent and 'SenderNodeID' relates to the node. Basically, there are two types of relationships between nodes via linkevents in the Commetrix data model: sender-receiver-relationships (e.g. emails) and thread-based relationships (e.g. discussion groups). The first type is stored in tables 'LinkeventSender' and 'LinkeventRecipient'. 'LinkeventSender' stores the relation of one or more senders or originators of a linkevent and 'LinkeventRecipient' stores the relation of one or more recipients or addressees of a linkevent. The second type of relationship, thread-based relationships, occur whenever a linkevent is originated by one or more node but not directly addressed to one or more other nodes, e.g. creating a new thread in a discussion group by posting a message. In this case, nodes interlink whenever one linkevent refers to another linkevent, e.g. whenever a post in a discussion group refers to another post. This reference is stored in the table 'LinkeventParent': column 'LinkeventID' refers to the referencing linkevent stored in table 'Linkevent' and column 'ParentLinkeventID' refers to the referenced linkevent, also stored in table 'Linkevent'. It is possible that a linkevent refers to several other linkevents. Combinations of both types of linkevent relationships are possible as well, especially if different types of linkevents are mixed, e.g. a discussion group allowing direct messages being send between authors. Some examples are shown in the table below:

Table 1: Examples of how to use the different types of Commetrix linkevent relationships.

Example	Network	Node	Linkevent	LinkeventSender	LinkeventRecipient	LinkeventParent
Email	X (general information)	X (senders and recipients)	X (emails)	1 (sender of the email)	N (one or more recipients of the email)	-
Discussion group	X (general information)	X (authors of a post)	X (posts)	1 (author of a post)	-	1 (one post references to another post)
Paper network with citation links	X (general information)	X (authors of a paper)	X (papers)	N (one or more authors of a paper)	-	N (one or more citations of other papers in each paper)
Wiki	X (general information)	X (author of a wiki article)	X (wiki articles)	N (one or more authors of an article)	-	N (an article references to other articles)

3.2 Table 'Network'

Table 'Network' stores the general data of a network. A database can consist of more than one network, thus each table links via column 'NetworkID' to the appropriate entry in table 'Network'. However, having large datasets we recommend to use a database for each network.

Network	
PK	<u>NetworkID</u>
	ElementTypeID Name Detail1 Detail2 Detail3 Detail4 Detail5

Figure 2: Commetrix data model – Table ‘Network’.

Columns ‘NetworkID’, ‘ElementTypeID’ and ‘Name’ are always required. ‘ElementTypeID’ has to be ‘00’ if the data is retrieved directly from the CMXProducer to create Commetrix network files. Columns ‘Detail1’, ‘Detail2’, ‘Detail3’, ‘Detail4’ and ‘Detail5’ contain additional information about the network. There are different ways how this information can be used in the CMXAnalyzer. This is stored in Table ‘NetworkDetailConfig’ which is explained later.

3.3 Table ‘Node’

Table ‘Node’ stores the information about the nodes in the network. A database can consist of more than one network, thus each table links via column ‘NetworkID’ to the appropriate entry in table ‘Network’. However, having large datasets we recommend to use a database for each network. The primary key in table ‘Node’ is the column ‘NodeID’. Thus, storing several networks in one database requires different IDs for each node.

Node	
PK	<u>NodeID</u>
	NetworkID Detail1 Detail2 Detail3 Detail4 Detail5 AliasID

Figure 3: Commetrix data model – Table ‘Node’.

Columns ‘NodeID’ and ‘NetworkID’ are always required. Columns ‘Detail1’, ‘Detail2’, ‘Detail3’, ‘Detail4’ and ‘Detail5’ contain additional information about each node. There are different ways how this information can be used in the CMXAnalyzer, e.g. size of a node, color of a node and node label. This is stored in Table ‘NodeDetailConfig’ which is explained later. Column ‘AliasID’ is not used at the moment.

For each node ‘Detail1’ is always required and has to be unique for each node, e.g. email address, username etc. If no unique value is available in the dataset naturally, the nodeID can be used.

3.4 Table ‘Linkevent’

Table ‘Linkevent’ stores the information about the linkevents in the network. A database can consist of more than one network, thus each table links via column ‘NetworkID’ to the appropriate entry in table ‘Network’. However, having large datasets we recommend to use a database for each network. The primary key in table ‘Linkevent’ is only the column ‘LinkeventID’. Thus, storing several networks in one database requires different IDs for each linkevent.

Linkevent	
PK	<u>LinkeventID</u>
	NetworkID LinkeventDate Subject Content Detail1 Detail2 Detail3 Detail4 Detail5

Figure 4: Commetrix data model – Table ‘Linkevent’.

Columns ‘LinkeventID’, ‘NetworkID’, ‘LinkeventDate’ are always required. ‘LinkeventDate’ is necessary for the time animation of the network in the CMXAnalyzer and the analysis of dynamic properties. If the data does not have a date naturally, then a default date has to be stored for each linkevent. For example, in MySQL the function SYSDATE() returns the current date.

Columns ‘Subject’ and ‘Content’ are used for the content analysis of the network. They contain text data. A short description or label of the content of the linkevent can optionally be stored in column ‘Subject’. Having an email dataset it would be the subject of the email with each email being a linkevent, having a paper dataset it would be the title of the paper with each paper being a linkevent. The full content of the linkevent (if available) should be stored in column ‘Content’. Having an email dataset it would be the body text of the email, having a paper dataset it would be the text of the paper itself. The default values should be an empty string (‘’) instead of ‘null’.

Linkevents are aggregated to links by the CMXProducer and visualized as edges in a graph by the CMXAnalyzer. Columns ‘Detail1’, ‘Detail2’, ‘Detail3’, ‘Detail4’ and ‘Detail5’ contain additional information about each linkevent. There are different ways how this information can be used in the CMXAnalyzer, e.g. size of a link, color of a link and link label. This is stored in Table ‘LinkeventDetailConfig’ which is explained later.

There are two different types how linkevents relate nodes. The basic concepts and the corresponding tables are already explained in section 3.1.

3.5 Tables ‘NetworkDetailConfig’, ‘NodeDetailConfig’ and ‘LinkeventDetailConfig’

There are three basic data tables in the Commetrix data model: ‘Network’, ‘Node’ and ‘Linkevent’. All three tables allow storing information about the data type in five different columns: ‘Detail1’, ‘Detail2’, ‘Detail3’, ‘Detail4’ and ‘Detail5’. This information can be used in the CMXAnalyzer to visualize the network. Generally, each detail can be used as label, size or color information. Network details can only be used as label as the network itself is not visualized but only its nodes and links. The corresponding tables ‘NetworkDetailConfig’, ‘NodedetailConfig’ and ‘LinkeventDetailConfig’ are used to specify each detail of the corresponding data type.

NetworkDetailConfig		NodeDetailConfig		LinkeventDetailConfig	
PK	Detail	PK	Detail	PK	Detail
	Label Description NetworkID isSize isColor isReadOnly isNumeric		Label Description NetworkID isSize isColor isReadOnly isNumeric		Label Description NetworkID isSize isColor isReadOnly isNumeric

Figure 5: Commetrix data model – Tables ‘NetworkDetailConfig’, ‘NodeDetailConfig’ and ‘LinkeventDetailConfig’.

Not all details have to be used, but only those further specified by some meta information in these tables will be included in the Commetrix network file by the CMXProducer.

The first column ‘Detail’ indicates the detail in use. The numeric values which can be used here are 1, 2, 3, 4, and 5 corresponding with the columns ‘Detail1’, ‘Detail2’, ‘Detail3’, ‘Detail4’ and ‘Detail5’. Using other values will be ignored by the CMXProducer.

Each detail has to be specified by its label, the name of the detail displayed in the CMXAnalyzer, and a brief description.

The column ‘NetworkID’ refers to the network for which the detail is stored as a database can contain more than one network with different details in use.

The column ‘isSize’ can be 0 (‘false’) or 1 (‘true’), indicating if the detail should be used as size information, e.g. node size or link width. Network details can only have the value 0 in this column.

The column ‘isColor’ can be 0 (‘false’) or 1 (‘true’), indicating if the detail should be used as color information, e.g. node color or link color. Network details can only have the value 0 in this column.

The column ‘isReadOnly’ can be 0 (‘false’) or 1 (‘true’), indicating if the detail should only be used as label information, e.g. node label, link label or network label. Network details can only have the value 1 in this column.

The column ‘isNumeric’ can be 0 (‘false’) or 1 (‘true’), indicating if the detail will contain only numeric values or alphanumeric values as well. The email of a node would be alphanumeric. Only numeric details can be used as size information. If a detail is used as color information, there are two different color scales for numeric and alphanumeric details. If an alphanumeric detail is incorrectly specified as numeric, the CMXProducer and/or the CMXAnalyzer will not work properly.

For each node ‘Detail1’ is always required and has to be unique for each node, e.g. email address, username etc. If no unique value is available in the dataset naturally, the nodeID can be used.

For example, a node detail could be the email address stored in ‘Detail1’ for each node in table ‘Node’ with the following entry in the ‘NodeDetailConfig’ table:

Table 2: Example entries for table ‘NodeDetailConfig’.

Detail1	Label	Description	NetworkID	isSize	isColor	isReadOnly	isNumeric
1	Email	Email address of a node	1	1	1	0	0

Linkevents are aggregated as links in the Commetrix network. Thus, linkevent details are also aggregated as link details. This is done automatically by the CMXAnalyzer. The CMXAnalyzer provides different functions how the linkevents shall be aggregated. At the moment these functions are: average, sum, minimum or maximum calculated from all linkevents of a link, value of the latest or oldest linkevent, or most or least often occurring value. The first four functions are only available for numeric details.

3.6 Tables 'Keyword', 'SubjectKeyword' and 'ContentKeyword'

The CMXAnalyzer allows an analysis of the content of the linkevents. Therefore, a preliminary text mining process has to be performed before creating a Commetrix network file by the CMXProducer. A very rough routine is available via the CMXProducer ('Import Keywords'). But for more sophisticated usage of the available features we recommend to employ an own text mining process on the Commetrix database. Here, the three tables 'Keyword', 'SubjectKeyword' and 'ContentKeyword' are important.

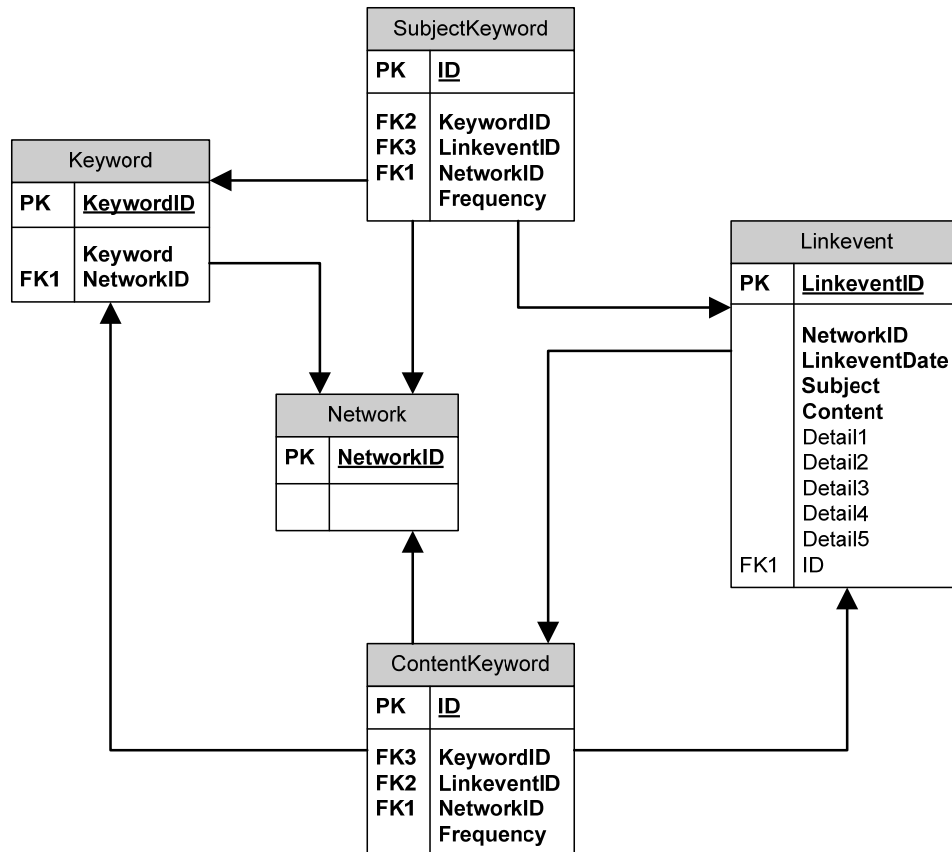


Figure 6: Commetrix data model – Tables 'Keyword', 'SubjectKeyword' and 'ContentKeyword' and how they relate to table 'Linkevent'.

The Table 'Keyword' stores all keywords occurring in the network (after employing the text mining process) with a unique ID (column 'KeywordID') and the keyword itself (column 'Keyword'). The column 'NetworkID' refers to the network to which the keyword belongs as a database can contain more than one network.

The occurrence of a keyword in the columns 'Subject' or 'Content' of a linkevent are stored in Table 'SubjectKeyword' and 'ContentKeyword', respectively. In both tables, the column 'LinkeventID' refers to the linkevent the keyword has been extracted from, the column 'KeywordID' refers to which keyword occurs and the column 'Frequency' to an occurrence frequency set calculated by the text mining process, e.g. how often the keyword is used in the subject or content of a linkevent. The column 'NetworkID' refers to the network to which the linkevent belongs as a database can contain more than one network.

3.7 Table 'Stopword'

The table 'Stopword' is used by the optional text mining process provided by the CMXProducer. All words stored in this table are not used as keywords. The column 'Word' contains the word itself, the column 'Lang' the language. At the moment only English words can be used, so the value has to be 'en'.

Stopword	
PK	Lang
PK	Word

Figure 7: Commetrix data model – Table 'Stopword'.

3.8 Tables 'Node_tmp', 'Linkevent_tmp', 'LinkeventSender_tmp', 'LinkeventRecipient_tmp', 'LinkeventParent_tmp', 'Keyword_tmp', 'SubjectKeyword_tmp' and 'ContentKeyword_tmp'

The CMXProducer allows to set several prefilter, e.g. to use the data stored in a linkevent detail as keyword or to ignore isolated nodes when creating a Commetrix network file. Whenever a file is created different filter options can be set. Thus, each network stored in a Commetrix database can be analyzed from different perspectives. The design and usage of the prefilter is explained in more detail in the CMXProducer manual. During the file creation process the tables 'Node_tmp', 'Linkevent_tmp', 'LinkeventSender_tmp', 'LinkeventRecipient_tmp', 'LinkeventParent_tmp', 'Keyword_tmp', 'SubjectKeyword_tmp' and 'ContentKeyword_tmp' are created and contain only the data used for creating the Commetrix network file. They are kept in the database until the CMXProducer is used again, to allow the user to examine the data if neccessary. The structure of these tables correspond to the original tables 'Node', 'Linkevent', 'LinkeventSender', 'LinkeventRecipient', 'LinkeventParent', 'Keyword', 'SubjectKeyword' and 'ContentKeyword'. These tables will be created by the CMXProducer, they are not included in the database scripts for creating an empty or exemplary Commetrix database.

4 Creating Networks from a Commetrix database

For a better understanding some small examples are given which illustrates how networking data can be stored in the Commetrix database and how networks will be visualized in the CMXAnalyzer based on the database entries.

4.1 Thread-based communication with only one node as sender

In this section a small example of a thread-based communication in a newsgroup is given. Columns or tables not in use are not included.

The basic data types in use are network, node and linkevent. These database tables are shown below:

Network

NetworkID	Name	ElementTypeID	Detail1
1	1	00	http://www.newsgroup-example.com

Linkevent

LinkeventID	NetworkID	LinkeventDate	Detail1
1	1	2007-12-12 11:19:01	post1
2	1	...	reply1.1
3	1	...	reply1.2
4	1	...	reply1.2.1
5	1	...	reply1.2.2
6	1	...	reply1.3
7	1	...	post2
8	1	...	post3
9	1	...	reply3.1

Node

NodeID	NetworkID	Detail1
1	1	chris89
2	1	martok
3	1	sunshine
4	1	claudé

Figure 8: Thread-based communication with only one node as sender – basic data tables.

Thread-based communication employs the tables LinkeventSender and LinkeventParent which relate nodes and linkevents. The corresponding database tables are shown below:

LinkeventSender

LinkeventID	SenderNodeID	NetworkID
1	1	1
2	2	1
3	3	1
4	2	1
5	2	1
6	3	1
7	1	1
8	3	1
9	1	1

LinkeventParent

LinkeventID	ParentLinkeventID	NetworkID
2	1	1
3	1	1
4	3	1
5	3	1
6	1	1
9	8	1

Figure 9: Thread-based communication with only one node as sender – linkevent relationships.

The details in use have to be specified in the corresponding tables:

NetworkDetailConfig

Detail	Label	Description	isSize	isColor	isReadOnly	isNumeric	NetworkID
1	url	url of original newsgroup data	0	0	1	0	1

NodeDetailConfig

Detail	Label	Description	isSize	isColor	isReadOnly	isNumeric	NetworkID
1	Username	Username of the author	0	1	0	0	1

LinkeventDetailConfig

Detail	Label	Description	isSize	isColor	isReadOnly	isNumeric	NetworkID
1	Description	Short description of the post	0	0	1	0	1

Figure 10: Thread-based communication with only one node as sender – Detail configuration.

The next figure shows how the linkevents and nodes relate to each other and how the resulting graph will look like.

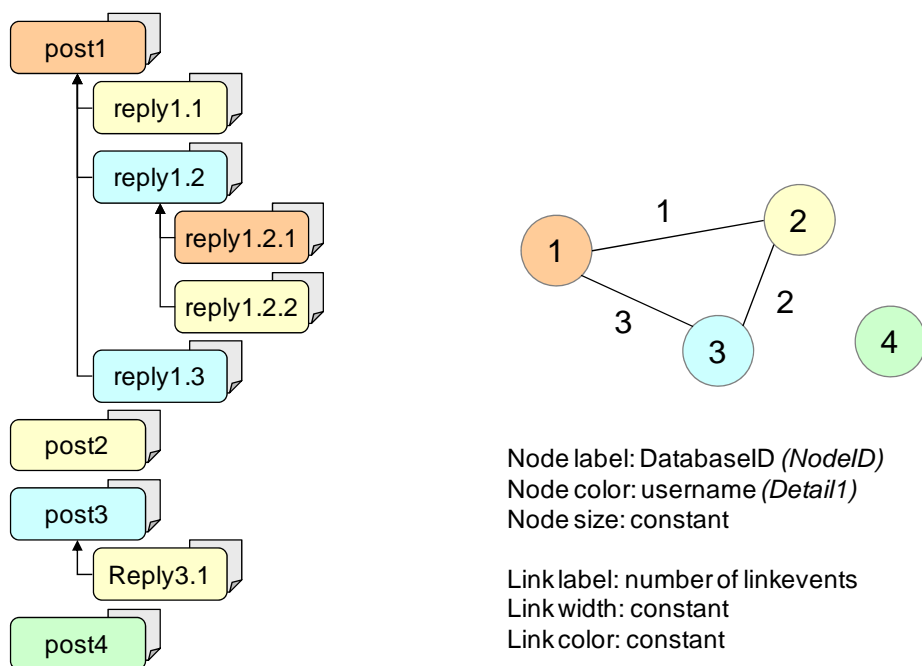


Figure 11: Thread-based communication with only one node as sender – visualization.

4.2 Thread-based communication with several nodes as senders

In this section a small example of a thread-based communication is given where several senders of a linkevent are allowed. Columns or tables not in use are not included.

The basic data types in use are network, node and linkevent. These database tables are shown below:

Network

NetworkID	Name	ElementTypeID	Detail1
1	1	00	http://www.newsgroup-example.com

Linkevent

LinkeventID	NetworkID	LinkeventDate	Detail1
1	1	2007-12-12 11:19:01	post1
2	1	...	reply1.1
3	1	...	reply1.2
4	1	...	reply1.2.1
5	1	...	reply1.2.2

Node

NodeID	NetworkID	Detail1
1	1	chris89
2	1	martok
3	1	sunshine

Figure 12: Thread-based communication with several nodes as senders – basic data tables.

Thread-based communication employs the tables LinkeventSender and LinkeventParent which relate nodes and linkevents. The corresponding database tables are shown below:

LinkeventSender

LinkeventID	SenderNodeID	NetworkID
1	1	1
1	3	1
2	2	1
3	3	1
4	1	1
5	1	1
5	2	1

LinkeventParent

LinkeventID	ParentLinkeventID	NetworkID
2	1	1
3	1	1
4	3	1
5	3	1

Figure 13: Thread-based communication with several nodes as senders – linkevent relationships.

The details in use have to be specified in the corresponding tables:

NetworkDetailConfig

Detail	Label	Description	isSize	isColor	isReadOnly	isNumeric	NetworkID
1	url	url of original newsgroup data	0	0	1	0	1

NodeDetailConfig

Detail	Label	Description	isSize	isColor	isReadOnly	isNumeric	NetworkID
1	Username	Username of the author	0	1	0	0	1

LinkeventDetailConfig

Detail	Label	Description	isSize	isColor	isReadOnly	isNumeric	NetworkID
1	Description	Short description of the post	0	0	1	0	1

Figure 14: Thread-based communication with several nodes as senders – Detail configuration.

The next figure shows how the linkevents and nodes relate to each other and how the resulting graph will look like.

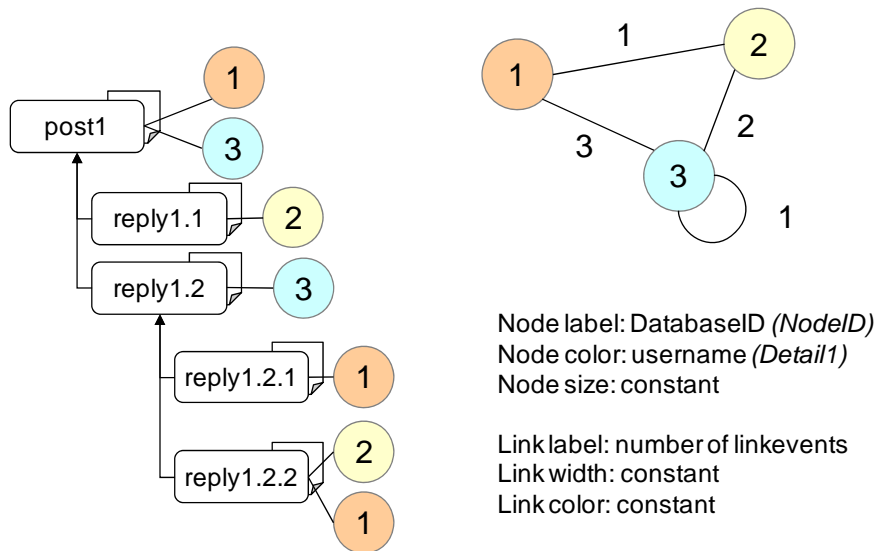


Figure 15: Thread-based communication with several nodes as senders – visualization.

4.3 Sender-receiver communication with only one node as sender

In this section a small example of a sender-receiver communication is given where only one sender of a linkevent is allowed. Columns or tables not in use are not included.

The basic data types in use are network, node and linkevent. These database tables are shown below:

Network

NetworkID	Name	ElementTypeID	Detail1
1	1	00	http://www.email-example.com

Linkevent

LinkeventID	NetworkID	LinkeventDate	Detail1
1	1	2007-12-12 11:19:01	email1
2	1	...	email2
3	1	...	email3
4	1	...	email4

Node

NodeID	NetworkID	Detail1
1	1	chris89@test.com
2	1	martok@test.com
3	1	sunshine@test.com
4	1	claudio@test.com

Figure 16: Sender-receiver communication with only one node as sender – Basic data tables.

Sender-receiver communication employs the tables LinkeventSender and LinkeventRecipient which relate nodes and linkevents. The corresponding database tables are shown below:

LinkeventSender

LinkeventID	SenderNodeID	NetworkID
1	1	1
2	2	1
3	1	1
4	3	1

LinkeventRecipient

RecipientNodeID	LinkeventID	NetworkID
2	1	1
3	1	1
3	2	1
2	3	1
1	3	1
4	4	1

Figure 17: Sender-receiver communication with only one node as sender – linkevent relationships.

The details in use have to be specified in the corresponding tables:

NetworkDetailConfig

Detail	Label	Description	isSize	isColor	isReadOnly	isNumeric	NetworkID
1	url	url of original email data	0	0	1	0	1

NodeDetailConfig

Detail	Label	Description	isSize	isColor	isReadOnly	isNumeric	NetworkID
1	Email	Email adress of the user	0	1	0	0	1

LinkeventDetailConfig

Detail	Label	Description	isSize	isColor	isReadOnly	isNumeric	NetworkID
1	Description	Short description of the email	0	0	1	0	1

Figure 18: Sender-receiver communication with only one node as sender – Detail configuration.

The next figure shows how the linkevents and nodes relate to each other and how the resulting graph will look like.

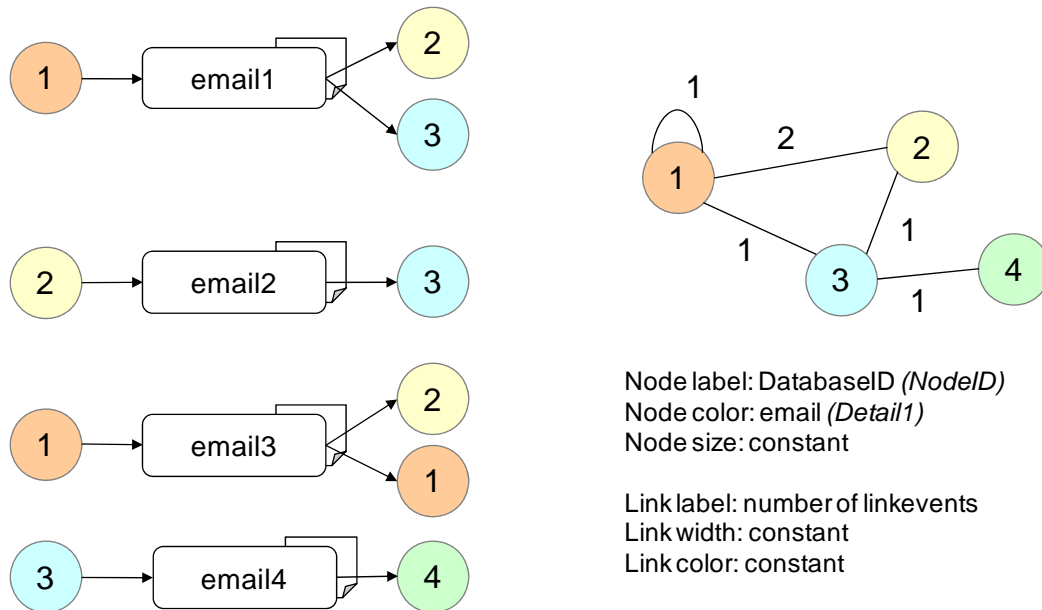


Figure 19: Sender-receiver communication with only one node as sender – visualization.

4.4 Sender-receiver communication with several nodes as senders

In this section a small example of a sender-receiver communication is given where several senders of a linkevent is allowed. Columns or tables not in use are not included.

The basic data types in use are network, node and linkevent. These database tables are shown below:

NetworkID	Name	ElementTypeID	Detail1
1	1	00	http://www.email-example.com

Linkevent

LinkeventID	NetworkID	LinkeventDate	Detail1
1	1	2007-12-12 11:19:01	email1
2	1	...	email2
3	1	...	email3
4	1	...	email4

Node

NodeID	NetworkID	Detail1
1	1	chris89@test.com
2	1	martok@test.com
3	1	sunshine@test.com
4	1	claudio@test.com

Figure 20: Sender-receiver communication with several nodes as senders – Basic data tables.

Sender-receiver communication employs the tables LinkeventSender and LinkeventRecipient which relate nodes and linkevents. The corresponding database tables are shown below:

LinkeventSender			LinkeventRecipient		
LinkeventID	SenderNodeID	NetworkID	RecipientNodeID	LinkeventID	NetworkID
1	1	1	2	1	1
1	4	1	3	1	1
2	1	1	3	2	1
2	2	1	2	3	1
3	1	1	1	3	1
4	1	1	4	4	1

Figure 21: Sender-receiver communication with several nodes as senders – linkevent relationships

The details in use have to be specified in the corresponding tables:

NetworkDetailConfig							
Detail	Label	Description	isSize	isColor	isReadOnly	isNumeric	NetworkID
1	url	url of original email data	0	0	1	0	1

NodeDetailConfig							
Detail	Label	Description	isSize	isColor	isReadOnly	isNumeric	NetworkID
1	Email	Email adress of the user	0	1	0	0	1

LinkeventDetailConfig							
Detail	Label	Description	isSize	isColor	isReadOnly	isNumeric	NetworkID
1	Description	Short description of the email	0	0	1	0	1

Figure 22: Sender-receiver communication with several nodes as senders – detail configuration.

The next figure shows how the linkevents and nodes relate to each other and how the resulting graph will look like.

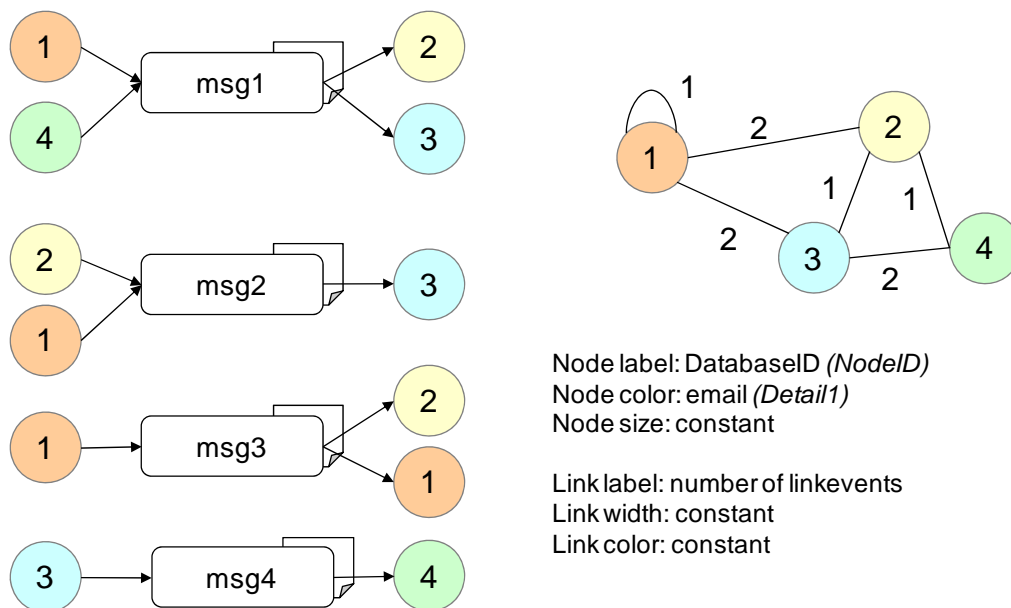


Figure 23: Sender-receiver communication with several nodes as senders – visualization.

4.5 Using linkevent-based communication for a co-authorship network

The sender-receiver communication with several nodes as senders can be used to model a co-authorship network, where one or more nodes are the author(s) of a paper. In this section a small example of such a network is given. Columns or tables not in use are not included.

The basic data types in use are network, node and linkevent. These database tables are shown below:

NetworkID	Name	ElementTypeID	Detail1
1	1	00	http://www.paper-database.com

Linkevent

LinkeventID	NetworkID	LinkeventDate	Detail1
1	1	2007-12-12 11:19:01	paper1
2	1	...	paper2
3	1	...	paper3
4	1	...	paper4

Node

NodeID	NetworkID	Detail1
1	1	chris89@test.com
2	1	martok@test.com
3	1	sunshine@test.com
4	1	claudio@test.com

Figure 24: Co-authorship network – basic data tables.

Sender-receiver communication employs the tables LinkeventSender and LinkeventRecipient which relate nodes and linkevents. The corresponding database tables are shown below:

LinkeventSender			LinkeventRecipient		
LinkeventID	NodeID	NetworkID	LinkeventID	RecipientNodeID	NetworkID
1	1	1	1	1	1
2	1	1	2	1	1
2	2	1	2	2	1
2	4	1	2	4	1
3	1	1	3	1	1
3	2	1	3	2	1
4	2	1	4	2	1
4	3	1	4	3	1

Figure 25: Co-authorship network – linkevent relationships.

The details in use have to be specified in the corresponding tables:

NetworkDetailConfig							
Detail	Label	Description	isSize	isColor	isReadOnly	isNumeric	NetworkID
1	url	url of original paper database	0	0	1	0	1

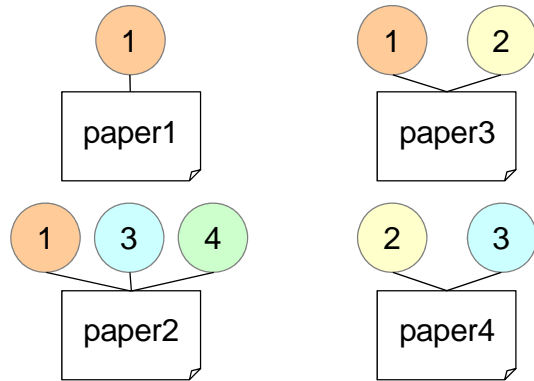
NodeDetailConfig							
Detail	Label	Description	isSize	isColor	isReadOnly	isNumeric	NetworkID
1	Email	Email adress of the author	0	1	0	0	1

LinkeventDetailConfig							
Detail	Label	Description	isSize	isColor	isReadOnly	isNumeric	NetworkID
1	Description	Short description of the paper	0	0	1	0	1

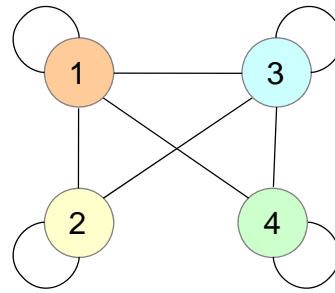
Figure 26: Co-authorship network – Detail configuration.

The next figure shows how the linkevents and nodes relate to each other and how the resulting graph will look like.

Co-authorships



Visualization as graph



Node label: DatabaseID (*NodeID*)
 Node color: email (*Detail1*)
 Node size: constant

Commetrix datamodel

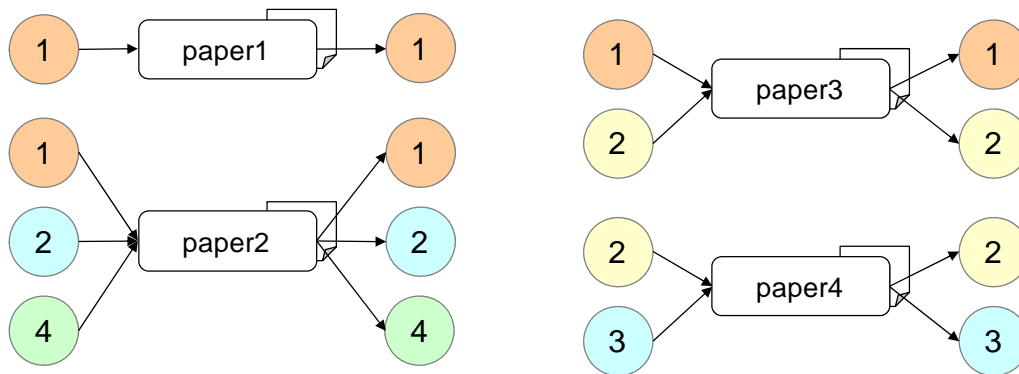


Figure 27: Co-authorship network – visualization.

4.6 Using sender-receiver and thread-based communication for a citation network

This section shows, how thread-based and sender-receiver communication can be used to establish different types of citation networks. Database tables are not included but only the visualization of the basic principles.

First, a citation network is established using only thread-based communication where the co-authorships are not included in the graph. The general idea and the Commetrix data model are illustrated in the figure below:

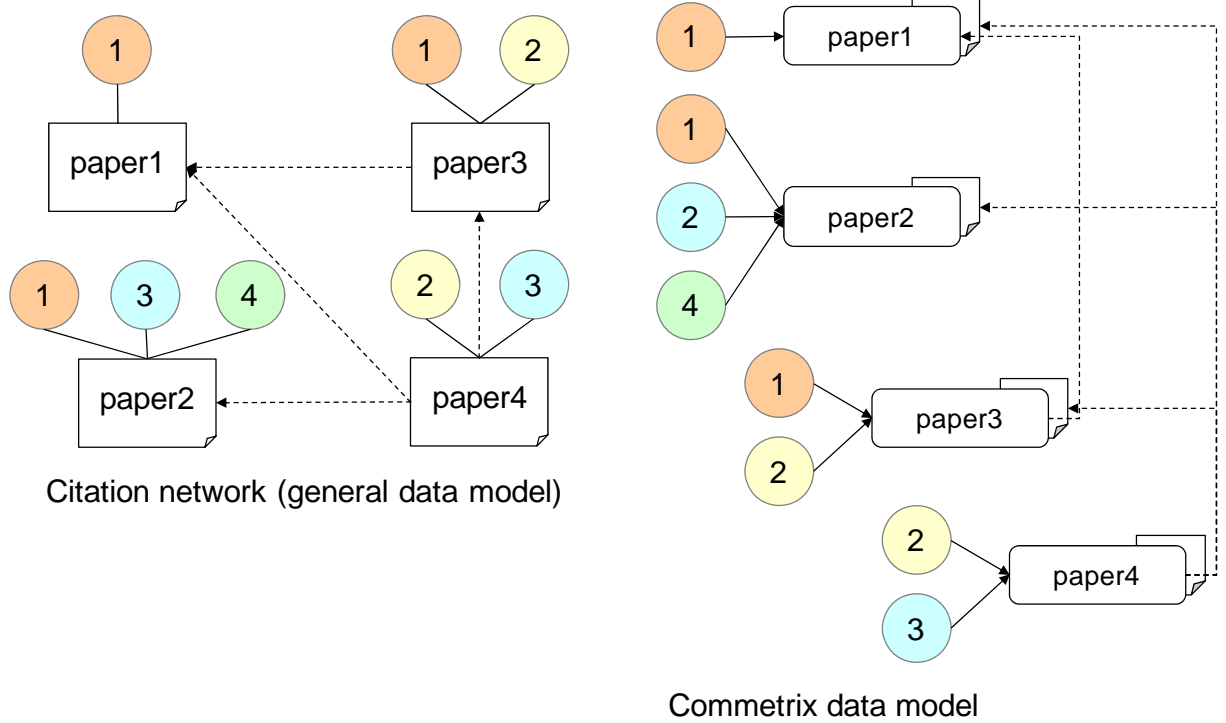


Figure 28: Co-citation network – visualization (Commetrix data model).

A link is established in the graph only if there is a citation between two nodes/papers. Co-authorship links are not included.

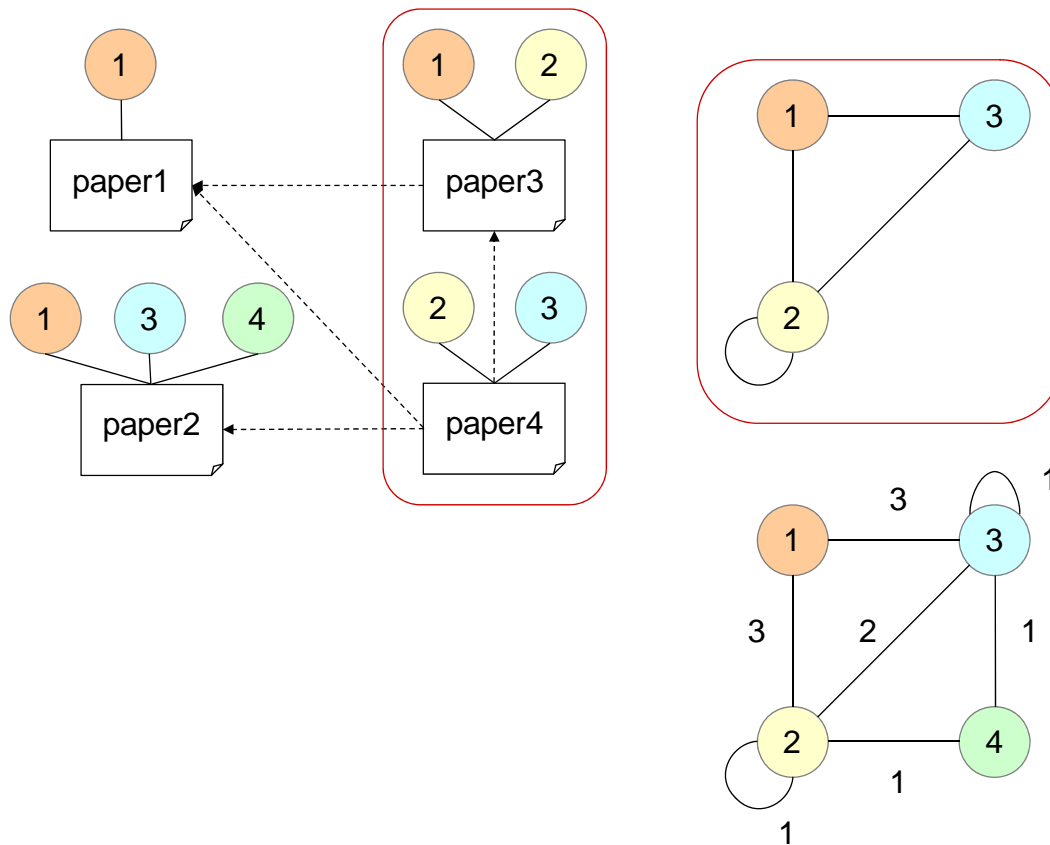


Figure 29: Co-citation network – visualization (graph).

Second, a citation network is established using sender-receiver as well as thread-based communication where the co-authorships are included in the graph. The general idea and the Commetrix data model are illustrated in the figure below:

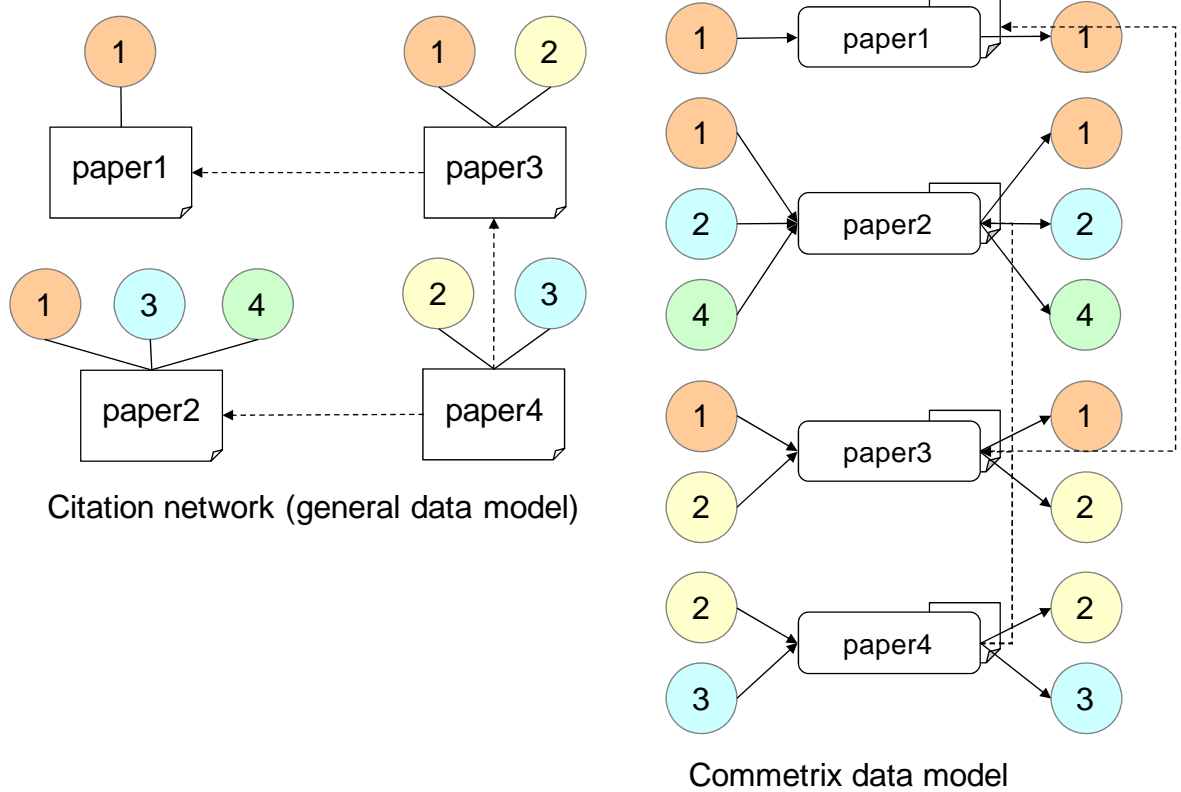


Figure 30: Co-authorship network with co-citation – visualization (Commetrix data model).

A link is established in the graph if there is a co-authorship between two nodes or a citation between two nodes/papers.

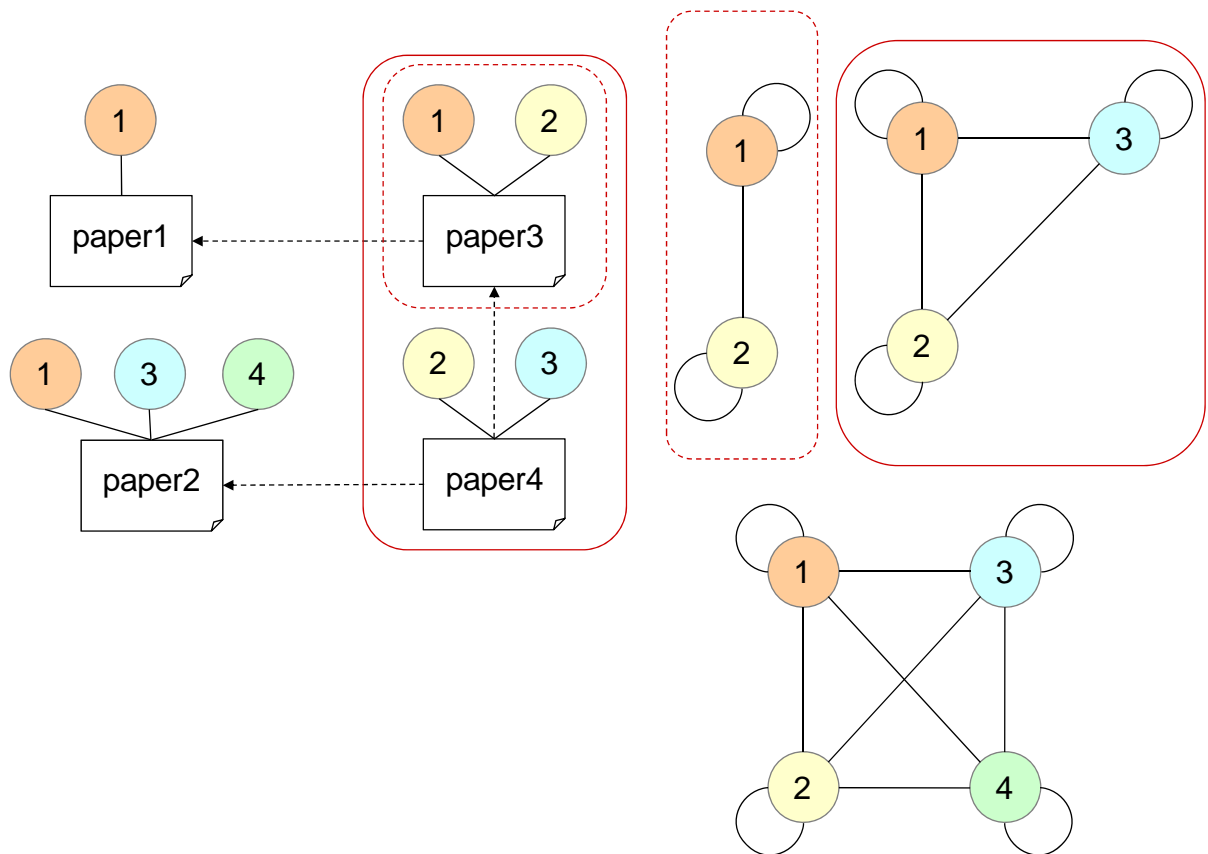


Figure 31: Co-authorship network with co-citation – visualization (graph).

5 Contact

If our Commetrix Framework Solution is of interest for you, we will be happy to arrange a license contract. Just contact us via: info@trilexis.com (preferred) or via the following address.



Trilexis Information Management GmbH
 Helmut-Kaeutner-Weg 5,
 13591 Berlin
 GERMANY
 ID: DE 29 039 00437
info@trilexis.com