Unified External Data Access Implementation in Formal Concept Analysis Research Toolbox

Alexey A. Neznanov, Andrey A. Parinov

National Research University Higher School of Economics,
20 Myasnitskaya Ulitsa, Moscow, 101000, Russia
ANeznanov@hse.ru, AParinov@hse.ru

Abstract. Formal Concept Analysis (FCA) provides mathematical models, methods and algorithms for data analysis. However, by now there is no easily available program system, which would provide data analyst with unified, intelligible and transparent access to various external data sources with large amount of heterogeneous data for subsequent FCA-based knowledge discovery. The lack of such tools complicates spreading FCA methods among big data analysts and miners of unstructured data. In this paper, we describe advances and new functionality in external data querying and preprocessing subsystems of Formal Concept Analysis Research Toolbox (FCART), which helps processing data of different types in a unified way.

Keywords: Formal Concept Analysis, Knowledge Extraction, Data Mining, Text Mining, Social Network Mining, Software.

1 Introduction

By now, mathematical models of Formal Concept Analysis (FCA) [1] are widely used for solving various problems of Knowledge Discovery and Artificial Intelligence [2,3]. Some systems use FCA ideas implicitly, by processing closed sets of attributes or objects. In this paper we will concentrate on explicit implementation of FCA methods as part of analyst's workflow in a software system. Three main problems here can be stated as follows.

1. How to generate suitable input data for FCA-based methods?
2. How to keep initial data properties and metadata while analyzing object-attribute representation by FCA-based methods?
3. How to combat high computational complexity of FCA-based methods in the context of an integral analyst's workflow?

Around the middle of the last decade, there were several successful implementations for transforming a relatively small formal context into a line diagram and computing implications and association rules. In [4] we have discussed well-known FCA-based tools, like ConExp [5], Conexp-clj [6], Galicia [7], Tockit [8],

ISBN 978-5-600-01454-1, National Research University Higher School of Economics, 2016.

ToscanaJ [9], Lattice Miner [10], OpenFCA [11], Coron [12,13], Cubist [14]. Most of the reviewed software tools are local applications that require initial data in the form of binary or many-valued context in one of the common formats (CSV, CXT or other). Thus, such programs can not be used on the stage of data gathering and preprocessing, but we should include input formats of those programs in the list of supported formats for future integration.

Formal Concept Analysis Research Toolbox (FCART) [15] supports iterative methodology of data mining and knowledge discovery. One of the goals of developing FCART is to create a system for handy analysis of heterogeneous data gathered from external data sources, e.g. SQL databases, NoSql databases and Social Network Services. FCART was successfully applied to analyzing data in medicine, criminalistics, sociology, and trend detection [3, 15].

In previous papers, we have described the system architecture, main workflow and stages of data extraction from various external sources. Here we would like to describe recent progress in the distributed version [16] of FCART and its Intermediate Data Storage (IDS) subsystem. This progress is mainly related to new functionality in data preprocessing.

2 Problems description

Data analysis is highly dependent on preprocessing, i.e., transformation from the source data format to the target data format, in which data are processed. An important functionality of any data analysis system is to support analyst in preprocessing transformations, making them transparent and easy.

2.1 A gap between FCA analytical artifacts and external data

From an analyst point of view, there is a gap between FCA analytical artifacts workflow and data and the legacy data. Fig. 1 illustrates this gap between "analyzable" and "external" data. It should be emphasized that it is not a gap between concrete data formats or access protocols, it is the gap in ways of thinking and knowledge representation.

The four main questions of object-attribute-value (or object-attribute) representation of data are trivial: 1) What are objects? 2) What are attributes? 3) How do we gather values of attributes? 4) How do we interpret values of attributes?

However, such questions bring into being a great many technological questions. For now, we can observe specific data preprocessing techniques of concrete data analysis projects. Can we propose fully unified approach? In general, the answer is *no*. However, we can try to adapt some common techniques for most popular classes of initial data formats and external data sources. On the one hand, we can see appearance of such terms as "Data Tidying" [17] for some "human readable" variants of ETL (Extract-Transform-Load) processes. On the other hand, there are continuous development of such monster software as Oracle Data Integrator Enterprise Edition [18] or less monstrous Microsoft PowerBI [19].

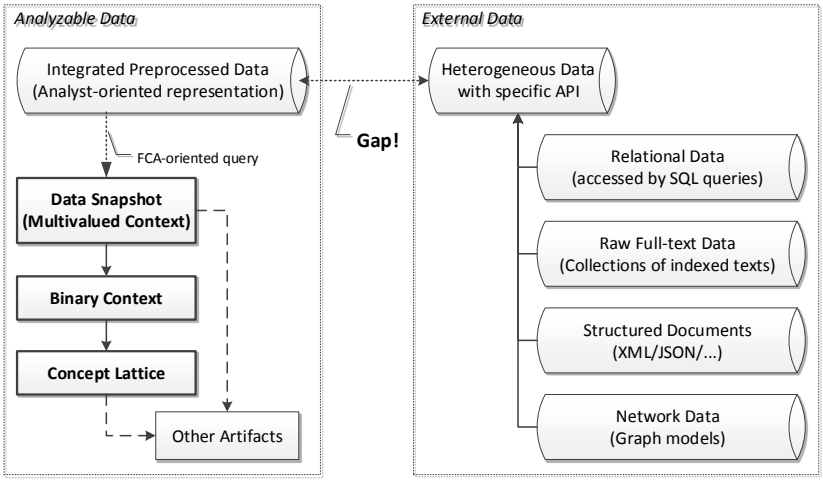

Fig. 1. The gap between "analyzable" and "external" data

FCA-based analytics tools impose additional requirements because of:

1. Basic FCA algorithms have very high computational complexity.
2. Big concept lattices are not suitable for interactive processing and visualizing.

There were many attempts to adapt FCA-based methods for complex tasks. For example, building Iceberg Lattices [20, 21, 22], visualizing other fragments of lattices, using incremental lattice construction algorithms.

2.2 A few comments about methodology

The core of the FCART supports knowledge discovery techniques, based on Formal Concept Analysis, clustering, multimodal clustering, pattern structures and others. From the analyst point of view, basic *FCA workflow* in FCART has four stages. On each stage, a user has the ability to import/export every artifact or add it to a report.

1. The filling Intermediate Data Storage (IDS) of FCART from various external SQL, XML/JSON and other data sources (querying external source is described by an *External Data Query Description* – EDQD). EDQD can be constructed by some visual External Data Browser (see later).
2. The loading a data snapshot from the IDS into an analytic session (a snapshot is described by a *Snapshot Profile*). A data snapshot is a data table with annotated structured and text attributes (a many-valued context) loaded in the system by accessing IDS.
3. The transforming a snapshot to a binary context (a transformation is described by a *Scaling Query*).
4. The building and visualizing formal concept lattice and other artifacts based on the binary context within an analytic session.

Later in this paper we will discuss mainly the first stage and using EDQDs. Hadley Wickham in [17] wrote: "there has been little research on how to make data cleaning as easy and effective as possible". The second and the third stages with example of Snapshot Profile construction were initially described in [23].

2.3 FCART architecture and the role of the IDS

The current distributed version of FCART consists of the following four parts:

1. *FCART AuthServer* for authentication and authorization, as well as integration of algorithmic and storage resources.
2. *FCART Intermediate Data Storage* (IDS) for storage and preprocessing (initial converting, indexing of text fields, etc.) of big datasets.
3. *FCART Thick Client* (Client) for interactive data processing and visualization in integrated graphical multi-document user interface.
4. *FCART Web-based solvers* (Web-Solvers) for implementing independent resource-intensive computations.

IDS plays important role in effectiveness of whole data analysis process because all data from external data storages, session data and intermediate analytic artifacts saves in IDS. All interaction between user and external data storages goes through the IDS. All interactions between Client, Web-Solvers and IDS go through a RESTful Web-API. The http-request to the IDS web-service constructed from two parts: prefix part and command part. Prefix part contains domain name and local path (e.g. http://zeus2.hse.ru:8444/). The command part describes what IDS has to do and represents some function of the Web-API. Using web-service commands, FCART client can query data from external data storages in uniform and efficient way.

Early we already have implemented populating IDS from external data sources, but now we extend the set of providers and improve data providers' EDQDs.

3 Worlds of data and data representation in IDS

Readers may have noticed that a simplest case of legacy data for object-attribute representation is *relational data* that meet the well-known conditions of E. Codd [24]. In this case we have virtually multivalued context. In the current state of Internet development we should distinguish at least the following types of data sources:

1. Relational data sources (directly queried by SQL).
2. NoSQL document collections (queried by XQuery or similar query languages).
3. Text collections with full-text index (queried by special full-text queries).
4. Social Network Services (with plenty of different access APIs).

3.1 Data integration problems and FCART Intermediate Data Storage

Documents are kept in many data formats (only ISO standards describe more then 400 formats, for example see [25]). After open data revolution [26] and Web infrastructure integration in Internet [27], most popular formats for information interchange are Comma Separate Values (CSV) [28], Extensible Markup Language (XML) [29] and JavaScript Object Notation (JSON) [30]. Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. The main goal of XML is to store metainformation with information itself. Hundreds of document formats using XML syntax have been developed, including RSS, Atom, SOAP, and XHTML. XML-based formats have become the default for many office-productivity tools, including Microsoft Office (Office Open XML), OpenOffice.org and LibreOffice (OpenDocument), and Apple's iWork. XML has also been employed as the base language for communication protocols, such as XMPP.

XML and its extensions have regularly been criticized for verbosity and complexity. JSON is lightweight alternative which focus on representing (serializing) programming language level objects with complex data structures rather than documents, which may contain both highly structured and relatively unstructured content. JSON is an open standard format that uses human-readable text to transmit data objects consisting of attribute–value pairs.

Traditional relational databases are not convenient for fast processing of big amounts of unstructured textual datasets with metadata. Document-oriented databases operating with documents in XML or JSON format are successfully used for storing, retrieving and managing big amounts of textual data in last decade. Both FCART IDS and FCART Client can handle XML and JSON documents as input format. XML format is complex and relatively hard to process at the same time. JSON format is more easy to use and lightweight. FCART uses JSON internally as a main format for data serialization and intercomponent communication.

3.2 Main terms and terminology problems

Preliminary problem of the discussed concepts is a terminological one. Table 1 illustrates the difference in approaches to defining terms for basic data-related concepts in SQL Servers (as stated in the SQL ISO Standard [31]), full-text indexing systems (as stated in the Elasticsearch reference [32]) and document-oriented NoSQL storages (as stated in the MongoDB reference [33]). One can look at term “Index” as a good example of polysemantic word. Graph-oriented databases use absolutely different terms for atomic elements (vertices, nodes, link, edges, arcs) and data structures, that reflect incidence, adjacency neighbourhood, etc.

In IDS we use data representation in form of “Databases” with hierarchical structure of “Collections” of JSON “Documents”. Each of the Documents may contains heterogeneous “Fields”. Each Collection can possess metadata, which describes structure of Documents and data types of Fields using JSON Schema [34].

It is very powerful approach, which gives an ability to validate Documents with compound data types.

Table 1. Real cases of different terms usage in popular data storages

	SQL Server	Elasticsearch	Mongo DB
1	Database (non normative)	**Index**	Database
2	Scheme	Mapping	--
3	Table	Type	Collection
4	**Index**	--	**Index**
5	Record/Row (Tuple)	Document (JSON)	Document (BSON)
6	Field/Column (Attribute)	Field	Field
7	Primary key	Document Id	`_id` field
8	Shard	Shard	Shard

4 External Data Queries in IDS

Extracting data is complicated by the fact that any Internet data source may have its own API. For example, we consider Social Network Services as a data source. That is why one needs mechanism to describe how data should be extracted, preprocessed and stored in IDS. For unified data access we developed External Data Query Description (EDQD) language. Each EDQD is a JSON formatted document. It aims to unify access to different data sources. By using EDQD FCART represents data from various data sources as a IDS Collection. There is no way to create a single query with fixed fields that would be with various data sources because each data source has its own set of functions, its own API. However, we developed the most common EDQD types and field set for mentioned above data sources types (Fig. 2).

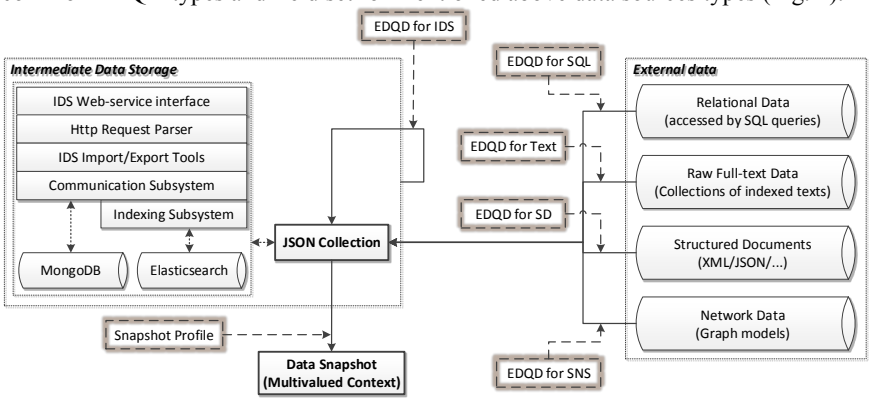

Fig. 2. The architecture of IDS and EDQDs for different data sources

Each field is a JSON object. An EDQD query includes next fields:

- ID – this field describes a unique identifier GUID.
- TYPE – this field describes type of data source. For now, FCART supports next types: “FS Folder”, “FS File”, “TSQL”, “REST”, “SOAP”, ”Facebook”, etc. Also EDQD with type “IDS” can refer to documents which are already stored in IDS.
- URI [27] – this field describes path to the data source. It is optional.
- CS – this field describes connection string [35]. It is optional.
- QUERY – this field describes query to the data source.
- TARGET – this field describes target data storage. For now, it’s can be set to the values – URI file path, “IDS”.
- TRANSFORMATION – this field describes type of source field, connection between source field and fields of target data storage; and all transformation such as scaling, indexing, etc.

This are the common fields for all EDQD queries. Below we have described specific for data sources EDQD queries.

Creating EDQD is a complex task, which needs visual tool for development. For now, External Data Browsers have been prepared to help user constructing EDQD for local JSON/XML files, unstructured text files and SQL data sources. Other types of EDQD can be created via direct JSON editing.

4.1 Query to a SQL data source

EDQD for a SQL data source is the most straightforward. For now, IDS supports connection to the Microsoft SQL 2014 (and its earlier versions) and Postgres 9.5.2 (and its earlier versions). EDQD for SQL has the following fields:

- “ID” – GUID (Globally Unique Identifier).
- “TYPE” – DBMS Type. Can be “TSQL” or “PS”.
- “URI” – This field is empty for that EDQD query type.
- “CS” – Connection String.
- “QUERY” – TSQL or PL-SQL query.
- “TRANSFORMATION” – For now it describes mapping a column name to a target field path in JSON document.

Example of EDQD for query to the instance of Microsoft SQL 2014:

```
{ “ID”: {6F9619FF-8B86-D011-B42D-00CF4FC964FF},
  ”TYPE”:”sql-server-2014”,
  “URI”:””,
  “CS”:”DataSource=190.190.200.100,1433;
Server=myServerName\myInstance;  Ininial  Catalog=myDataBase;  User
ID=myUsername; Password=myPassword;”,
  “QUERY”:”SELECT column_name FROM table1 INNER JOIN table2 ON
table1.column_name=table2.column_name;”
  “TRANSFORMATION”: { “field”:{
    “name”: “name”,
    “target_field”:”user_name”}
}
```

4.2 Query to unstructured text files

EDQD for Text (a collection of files with unstructured text) provides ability to extract and transform data from unstructured texts. To analyze data from unstructured data file we need to create an inverted index. Using inverted index reduces searching time for every text word.

The inverted index is a central component of indexing search engine. A goal of a search engine implementation is to optimize the speed of the query: find the documents where word X occurs. Once a forward index is developed, which stores lists of words per document, it is next inverted to develop an inverted index. Querying the forward index would require sequential iteration through each document and each word to verify a matching document. The time, memory, and processing resources to perform such a query are not always technically realistic. Instead of listing the words per document in the forward index, the inverted index data structure is developed which lists the documents per word [36].

To create inverted index, we use full-text search engine. For now, there are many full-text search engines, which provides rapid search, complicated query language and REST interface. Solr [37] and Elasticsearch [32] are the most powerful and popular search engines for now. In the previous paper [38] we described detailed comparison of Solr and Elasticsearch as basis for implementing full-text manipulating part of IDS. In the paper we showed speed advantage of Elasticsearch in situation of indexing and inserting data at the same time. It's important to search text data because unstructured text is often a part of other data types, e.g., structured documents (CSV, JSON, XML), documents extracted from social network services (user information, posts).

Initial sets of automatically extracted keywords may be very big. We can have additional instruments for such sparse contexts with many uniform attributes like sorting and searching attributes (Fig. 3) or analyzing attributes usage statistics. But more proper way to generate initial context is using adjustable query.

Example of EDQD for a query to a folder with text files:

```
{ "ID": {6F9619FF-8B86-D011-B42D-00CF4FC964FF},
  "TYPE":"FS folder"
  "URI":"file://localhost/c|/source/"
  "CS":""
  "QUERY":""
  "TARGET":" file://localhost/c|/target"
  "TRANSFORMATION": {
    "field": {
    "name": ""
    "target_field":"body",
    "type":"text",
    "indexing": True}
}
```

Field "Transformation" is the most interesting part of EDQD for a local text-files folder. "Name" refers to a field of result IDS document is affected. "Target_field" describes name in IDS document. "Type" describes type of the source field. The value of the EDQD field "Type" determines operations and transformations which are applicable to a document field. By now, FCART supports an indexing operation on the "text" type. By default, the value of "Indexing" field is False.

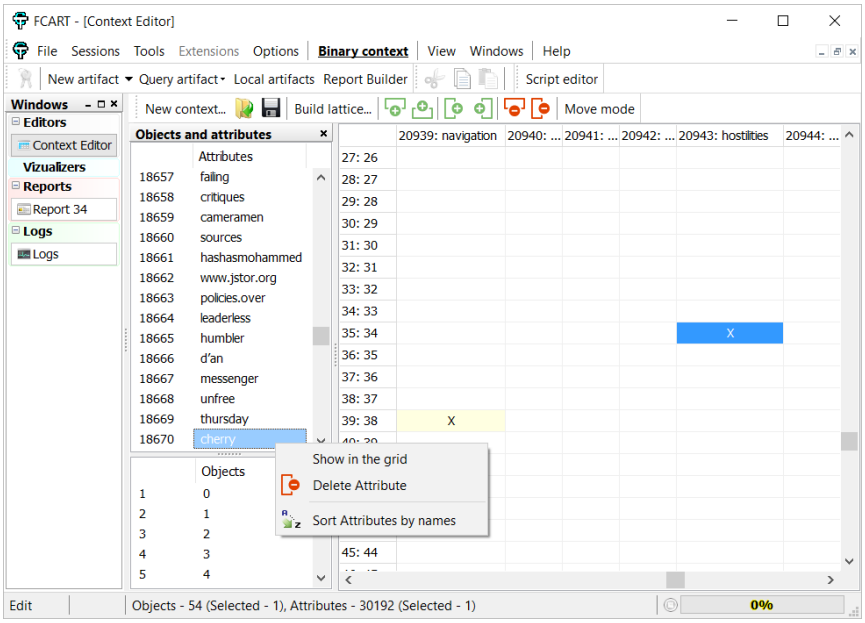

Fig. 3. Example of sparse context for all keywords in 54 documents – 30192 attributes!

4.3 Query to an XML, JSON or CSV data file

EDQD for a collection of XML [29], JSON [30] or CSV [28] data files is similar to EDQD for the text file. In case of XML one should specify path to the source document field in the EDQD field "source_path" according to the XPath standard [39]. In case of JSON one should use the JSONPath draft [40]. In the case of CSV one should use fragment identifier according RFC-7111 [41].

Example of EDQD for a query to a folder with collection of JSON data files:

```
{ "ID":
  "TYPE": "JSON Folder"
  "URI": "file://localhost/c|/source/"
  "CS ": ""
  "QUERY": ""
  "TARGET": "file://localhost/c|/target"
  "TRANSFORMATION": {
    "field": {
    "name": "body"
    "source_path": "/store/book/title"
    "target_field": "/body" }
}
```

4.4 EDQD query to a web-service

EDQD for web-service interface provides ability to extract data from web-service. In the current version FCART supports REST and SOAP interfaces. EDQD for web-service has the following fields:

- "ID" – GUID (Globally Unique Identifier).
- "TYPE" – Web-service type. Can be "REST" or "SOAP".
- "URI" – URI of web-service.
- "CS" - This field is empty.
- "QUERY" – JSON document which contains query.
- "TRANSFORMATION" – JSON document which describes field mapping.

Example of EDQD query to an Elasticsearch REST interface:

```
{ "ID": {6F9619FF-8B86-D011-B42D-00CF4FC964FF},
  "TYPE":"REST",
  "URI":"http://elasticsearch:1234/index_name/mapping_name/",
  "CS": "",
  "QUERY": "{
    "query": {
    "bool": {
    "must": [
      {"match":{"address": "mill" }},
      {"match":{"address": "lane" }}
    ] }}}",
  "TRANSFORMATION": "{
    "field":{
    "target_field": "body",
    "indexing": True}"
}
```

REST interfaces can iterate set of elements, which are returned by query. Query field contains JSON document written on Elasticsearch query language (https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html).

Transformation field contains JSON document, which describes target field and preprocessing operation. The current version of FCART supports only indexing operation.

4.5 EDQD query to a Social Network Service

Social networks services have special API types. FCART processes the most common part of social networks services analysis. EDQD represents a user profile of Social Network Service as a hierarchical JSON document that has next fields:

```
{"user":{
     "id":".."
     "path": ".."
     "user_info": ".."
     "friend:[…]
     }
"post":{
     "time":
```

```
      "body: ".."
      "title" ".."
      "tags":[ ".."],…, ".."]}
}
```

Besides full-text queries, analyst can query neighborhood of a person, e.g. friends or colleagues. In the current version FCART server supports connection to Livejournal, Twitter, and Facebook. Example of extracting posts from third neighborhood layer of the person:

```
{ "ID": {6F9619FF-8B86-D011-B42D-00CF4FC964FF},
  "TYPE": "facebook"
  "URI": "https://www.facebook.com/someuser/"
  "CS ": ""
  "QUERY": {
    "path": "friend/post/body",
    "layer": "3",
    "BEGIN": "2005-08-09T18:31:42-03",
    "END":"",
    "COUNT":100 },
  "TARGET":"IDS"
  "TRANSFORMATION": {
    "field":{
    "type":"post",
    "target_field":"body",
    "indexing": True }
}
```

5 Discussion and future work

In this paper, main problems of external data access in FCA-based analytics software were addressed and some real cases were examined while implementing new functionality in the FCART system. The demo version of FCART client is available at https://cs.hse.ru/en/ai/issa/proj_fcart and the test version of the IDS Web-service is available at http://zeus2.hse.ru:8444.

For FCA-based data analysis fundamental requirements for software are as follows:

1. The ability to merge heterogeneous data sources in a query to external data.
2. The ability to cache frequent queries.
3. The automatic populating of query metadata.
4. The support of many formats of local data files to communicate with other software tools easily.
5. The support of apriori prescribed constraints on FCA algorithms and visualization schemes.
6. The availability of common and special "quick and dirty" methods of query result visualization with low computational complexity.

When prototyping clinical decision support system components, we have realized the importance of having local and web-based versions of the preprocessing tools. So unification of external data access tools is the first step in satisfying informal analysts'

wishes. We also understand importance of other subsystems, including efficient data transformation algorithms, dashboards, etc. However, without unified and reproducible access to initial data no one can build real data analysis workflow.

Improved mechanisms of query data work faster, more intelligible and provide necessary information to data analyst. The next steps in our development process are adding new External Data Browsers, increasing efficiency of EDQD processing and standardizing new API for running Web-Solvers inside IDS instead of Client.

Acknowledgments This work was carried out by the authors within the project "Mining Data with Complex Structure and Semantic Technologies" supported by the Basic Research Program of the National Research University Higher School of Economics.

References

1. Ganter, B., Wille, R. Formal Concept Analysis: Mathematical Foundations, Springer, 1999.
2. Poelmans, J., Kuznetsov, S.O., Ignatov, D.I., Dedene, G. Formal Concept Analysis in knowledge processing: A survey on models and techniques. Expert Systems with Applications, 40(16), 2013 pp. 6601-6623.
3. Poelmans, J., Ignatov, D.I., Kuznetsov, S.O., Dedene, G. Formal Concept Analysis in knowledge processing: A survey on applications. Expert Systems with Applications, 40(16), 2013, pp. 6538-6560.
4. Neznanov, A.A., Parinov, A.A. About Universality and Flexibility of FCA-based Software Tools. Proceedings of the 3rd International Workshop "What can FCA do for Artificial Intelligence?", 2014, pp. 59-65.
5. Yevtushenko, S.A. System of data analysis "Concept Explorer" (In Russian). 7th National Conference on Artificial Intelligence (KII-2000), Russia, 2000, pp. 127-134.
6. Conexp-clj (http://daniel.kxpq.de/math/conexp-clj)
7. Valtchev, P., Grosser, D., Roume, C. Hacene, M.R. GALICIA: an open platform for lattices, in Using Conceptual Structures // Contributions to the 11th International Conference on Conceptual Structures (ICCS'03), 2003, pp. 241-254.
8. Tockit: Framework for Conceptual Knowledge Processing (http://www.tockit.org)
9. Becker, P., Hereth, J., Stumme, G. ToscanaJ: An Open Source Tool for Qualitative Data Analysis. Workshop FCAKDD of the 15th European Conference on Artificial Intelligence (ECAI-2002). Lyon, France, 2002.
10. Lahcen, B., Kwuida, L. Lattice Miner: A Tool for Concept Lattice Construction and Exploration. Suplementary Proceeding of International Conference on Formal Concept Analysis (ICFCA'10), 2010.
11. Borza, P.V., Sabou, O., Sacarea, C. OpenFCA, an open source formal concept analysis toolbox. IEEE International Conference on Automation Quality and Testing Robotics (AQTR), 2010, pp. 1-5.
12. Szathmary, L. The Coron Data Mining Platform (http://coron.loria.fr)
13. Szathmary, L., Napoli, A., Kuznetsov, S.O. ZART: A Multifunctional Itemset Mining Algorithm. 5th International Conference on Concept Lattices and Their Applications (CLA'07), pp.26-37.
14. Cubist Project (http://www.cubist-project.eu)

15. Neznanov, A., Ilvovsky, D., Kuznetsov, S. FCART: A New FCA-based System for Data Analysis and Knowledge Discovery. Contributions to the 11th International Conference on Formal Concept Analysis, Dresden, 2013, pp. 31-44.
16. Neznanov, A.A., Parinov, A.A. Distributed architecture of data analysis system based on formal concept analysis approach. Studies in Computational Intelligence, 616, 2016, pp. 265-271.
17. Wickham, H. Tidy Data. Journal of Statistical Software, 59 (10), 2014, pp. 1-23.
18. Oracle Data Integrator Enterprise Edition (http://www.oracle.com/us/products/middleware/data-integration/enterprise-edition/overview/index.html)
19. Microsoft PowerBI (http://powerbi.microsoft.com)
20. Rouane, M.H., Nehme, K., Valtchev, P., Godin, R. On-line maintenance of iceberg concept lattices. ICCS-2004, 2004.
21. Szathmary, L., Valtchev, P., Napoli, A., Godin, R., Boc, A., Makarenkov, V. Fast Mining of Iceberg Lattices: A Modular Approach Using Generators. CLA-2011, 2011, pp. 191-206.
22. Smith, D.T. A Formal Concept Analysis Approach to Data Mining: The QuICL Algorithm for Fast Iceberg Lattice Construction. Computer and Inf. Science, 7(1), 2014, pp. 10-32.
23. Neznanov, A.A., Kuznetsov, S.O. Information Retrieval and Knowledge Discovery with FCART. Proceedings of the Workshop Formal Concept Analysis Meets Information Retrieval (FCAIR 2013), CEUR-977, 2013, pp. 74-82.
24. Codd, E. F. Further Normalization of the Data Base Relational Model. Data Base Systems: Courant Computer Science Symposia Series 6, Prentice-Hall, 1972, pp. 33-64.
25. ISO Standards catalogue 35.040: Character sets and information coding (http://iso.org/iso/products/standards/catalogue_ics_browse.htm?ICS1=35&ICS2=040)
26. Open Data Watch (http://opendatawatch.com)
27. RFC-3986: Uniform Resource Identifier (URI): Generic Syntax (https://tools.ietf.org/html/rfc3986)
28. RFC-4180: Common Format and MIME Type for Comma-Separated Values (CSV) Files (http://tools.ietf.org/html/rfc4180)The Connection Strings Reference (http://www.connectionstrings.com)
29. Extensible Markup Language (XML) 1.0 (Fifth Edition), W3C Recommendation, 2008 (https://www.w3.org/TR/REC-xml)
30. Standard ECMA-404: The JSON Data Interchange Format, 2013 (http://www.ecma-international.org/publications/standards/Ecma-404.htm)
31. ISO/IEC 9075-2:2011 Information technology - Database languages - SQL - Part 2: Foundation (https://www.iso.org/obp/ui/#iso:std:iso-iec:9075:-2:ed-4:v1:en)
32. Elasticsearch Reference (https://www.elastic.co/guide/en/elasticsearch/reference)
33. MongoDB Reference Glossary (https://docs.mongodb.org/manual/reference/glossary)
34. The home of JSON Schema (http://json-schema.org)
35. The Connection Strings Reference (http://www.connectionstrings.com)
36. The inverted index description (https://en.wikipedia.org/wiki/Inverted_index)
37. Apache Solr (http://lucene.apache.org/solr)
38. Neznanov, A.A. Parinov, A.A. Full-text Search in Intermediate Data Storage of FCART. RuZA2015 Workshop, 2015. (http://ceur-ws.org/Vol-1552/paper7.pdf)
39. XPath Syntax (http://www.w3schools.com/xsl/xpath_syntax.asp)
40. JSONPath (http://goessner.net/articles/JsonPath)
41. RFC-7111: URI Fragment Identifiers for the text/csv Media Type (https://tools.ietf.org/html/rfc7111)