ACADEMA Ltd.
Tržaška cesta 132
SI – 1000 Ljubljana
Phone: +386 1 423 32 82
Email: contact@academa.si
Web: https://www.academa.si

Ljubljana, May 2021 (Version: 1.1)

ClueBoomBus: Distributed Data Warehouse/Data Marts of Streamed Linked Data as Knowledge Base System

Introduction

Data warehouse is considered a core component of business intelligence. It's a centralized system used for reporting and data analysis. It collects relevant current and historical data, from one or more disparate sources, before onwards distribution to data marts. These marts then structure and index data in forms appropriate for the desired consumption pattern. We found out that a similar model could work well for linked data. However, we did not find industry solutions that would serve this need well except the concept of Thomson Reuters CM-Well, particularly optimized for incremental flow of linked data, and that allows managing large volumes of linked data and warehousing its knowledge graph.

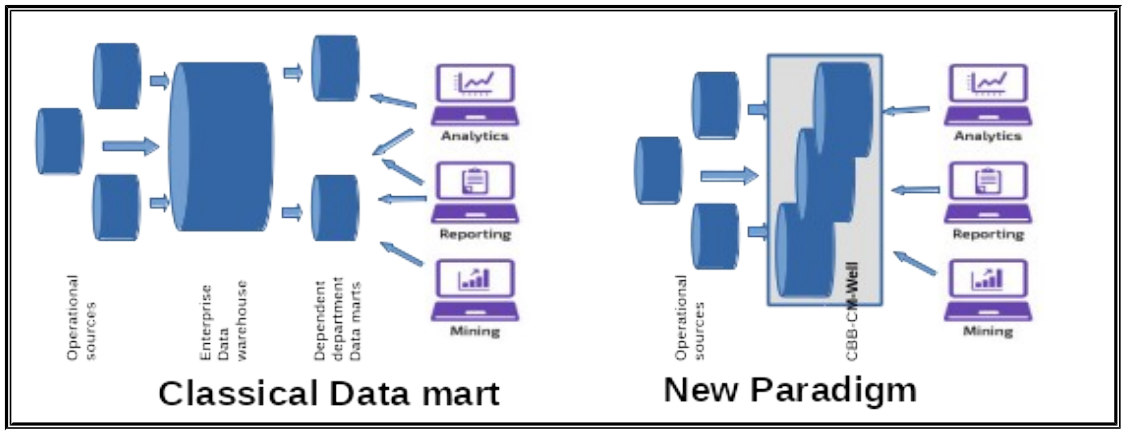

Fig.1

We introduce CBB: an application for managing large volumes of linked data. "The focus of CBB is on content management and distribution: higher-level functions, such as reasoning, are left to other systems, but will be integrated in the next step".

During use, we consider several shortcomings, so we internally redesign and add functionalities to CM-Well (mayor changes: redesign for streaming purpose, r/w permissions, sequencing, REST functionalities, integration of Shacl (Shapes Constraint Language), a language for validating RDF graphs against a set of conditions, etc.)

In our focus is to implement reasoning in the triggering process and to improve (enrich) data facts in different contexts (knowledge representation).

While CM-Well includes a browser based user interface, we designed active matrix (looks like spreadsheet, drag and drop functionality) in browser which works asynchronous. Typically most client interactions are via REST API.

Architecture

ClueBoomBus (CBB) is based on a clustered architecture (Fig. 2). The design of the solution consists of a well tested integration of a number of open source packages including CM-Well, Keycloak, Akka, Cassandra/FoudationDB, ElasticSearch, Jena, Shacl and Kafka.

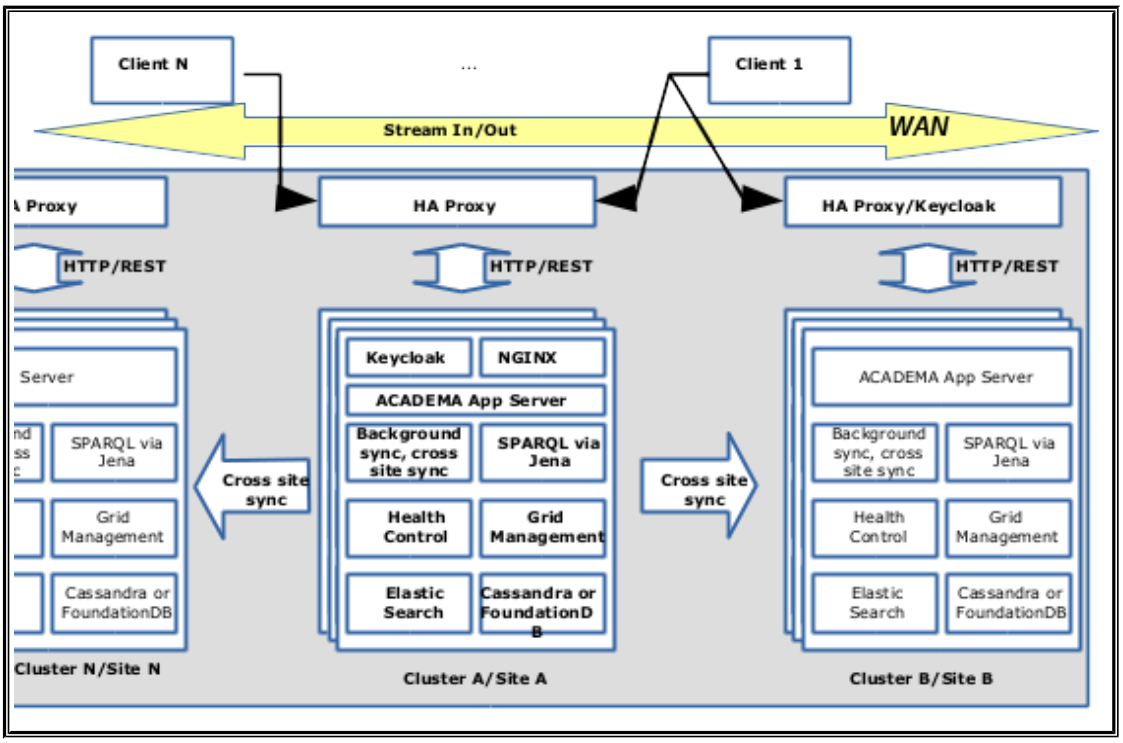

Fig.2

Each node in the cluster has the same configuration and runs a set of processes with no single point of failure. Singleton control roles are moved between nodes on failure, implemented on a "self healing" principle. The majority of the application is written in the Scala language.

Information Objects (immutable Infotons)

Individual triples are grouped by subject to form an "Infoton": the basic unit of storage. These information objects are stored in Cassandra/FoundationDB and invert indexed by ElasticSearch. While writes require a full rewrite of the infoton, reads by subject can be served from one node and return all triples for the subject. This grouping of triples forms a fundamental design trade-off between read and write performance but has been found to offer good horizontal scale performance: "on production cluster, API response time for all information on a subject is around 10ms".

Every Infoton write is treated as immutable; this approach permits eventually consistent replication across distributed data centers with secondary instances subscribing to a change log and applying writes locally. Depending on read/write trade-off, all slaves can replicate from a single master or slaves can form a branching chain.

Application uses an information unit called an "infoton". An infoton is the set of all the triples associated with a single subject entity. Each infoton:

- Represents a single entity or object.
- Has a URI, which serves as an ID of the object (which is the subject of the collection of triples the infoton contains), and which also serves as the infoton's reference address.
- Can have several attributes (the "objects" in the infoton's triples). Attributes can be either literal values or links to other infotons. Attribute types have namespaces, and can refer to external ontologies.

- Has one current version, pointed to by the URI, and optionally several "historical" versions, each with its own Universally Unique Identifier (UUID).
- The data is still logically structured as a set of separate triples, but the infoton is physically stored together with the set of its predicate-subject relationships

By "flattening" the physical model in this way, system is able to more easily scale to very large numbers of triples, as it becomes easier to store the data horizontally across many more servers. This model is a combination of a traditional key-value database approach and a document-based approach (where the infoton is the "document").

Infotons can have the following types:

- Object: the most common type of infoton, described above; contains a collection of triples.
- File: an all-purpose text or binary file (PDF, GRIB, movie, …) and contains a collection of triples.
- Link: points to another infoton (similar to a redirected URL).
- Directory: an infoton can be just a container for other infotons. Directory infotons may (but don't have to) have user-defined field values. Entity infotons and Directory infotons "look the same". The only difference is that Directory infotons can be created implicitly by creating another infoton with a path composed of '/'-separated parent directories.

Deployment

While we have found the application capable of balancing read and write traffic, in most applications consumers wrap the application with a data-mart implemented to meet the specific needs of their application. Depending on use case, this mart might be as simple as a caching reverse HTTP proxy to reduce read load or deployment of a separate search cluster which structures data to meet required performance targets.

Key features

One of the motivating factors for the development was our perception that key requirements were not met by other tools; either open or closed source. Integrating our own solution permitted us to focus on key differentiating features.

Distributed Data Warehouse

The data warehouse is a critical system for business intelligence and digital transformation. Migration to the cloud has made data warehouses easier to deploy and use. However, all centralized data warehouses suffer from the same architectural flaw – having to transfer large volumes of data across network.

With distributed data warehouse architecture, we eliminate latency, cost and privacy concerns by not having to move data from one location to another, and to knock down the data silos within customer's organization. It's a future proof of the customer investment with a system designed to handle changing data requirements without any redesign.

It's easy to choose the locations where the customer wish to warehouse data – combining on-premise, public cloud and edge deployments. In the cloud using public data centers such as those from Amazon AWS, Microsoft Azure, Google Cloud and Oracle Cloud and/or at the edge using physical or virtualized hardware in places like offices, factories, hospitals, stores and banks.

Warehouse data in the locations where it works is the best for the user. Application seamlessly ties everything together as one logical warehouse and avoid locking into any data center location, cloud-provider or hardware type.

Subscription by query

The API layer includes a REST based API for boolean querying by predicate. Any query can be used as a (server-side stateless) subscription; returning a time ordered sequence of Infotons matching the query.

Each stream API call returns a continuation token in the HTTP response headers. When passed in a subsequent request, this token is used as a point-in-time marker with the stream continuing from the end of the previous response. Additionally, an optional parameter will instruct application to include tombstones for deleted data, effectively treating each query as a FIFO queue of content events.

This feature is key to the "information in motion" design of application permitting downstream stores to subscribe to relevant subsets of data defined by query.

SPARQL Support

Currently, application supports two types of SPARQL query: sub and full graph. When querying, the sub-graph solution consists of two sub-steps: 1) one or more queries, routed to ElasticSearch, are first used to select a candidate set of nodes from our data; 2) Such candidate nodes are then loaded to a Jena triple store on a single machine for SPARQL execution. We are also experimenting with a second form which builds an execution plan for input ElasticSearch queries based on data statistics relevant to the SPARQL query. Both solutions are limited to the underlying characteristics of ElasticSearch, such maximum results window of 10,000. Hence, wide ranging queries can miss triples of relevance.

Triggers

Much of the data we load to the application is highly normalized and often requires simplification to ease downstream consumption. The trigger mechanism permits a sensor to be created, based on a query, which on new data invokes a SPARQL CONSTRUCT statement. This then generates new predicates.

Data Flow - Streaming In/Out

As we have an all-purpose text or binary file as a content of infoton, we redesign the whole process of reading and writing of content directly to/from datastore without rewriting of the content. Memory based rewrites limit the capabilities, so now we can stream-in/stream-out complete movie, GRIB or backup file.

Shacl Integration

We integrate SHACL (Shapes Constraint Language), a language for validating RDF graphs against a set of conditions. These conditions are provided as shapes and other constructs expressed in the form of an RDF graph. RDF graphs that are used in this manner are called "shapes graphs" in SHACL and the RDF graphs that are validated against a shapes graph are called "data graphs" (Infotons). As SHACL shape graphs are used to validate that data graphs satisfy a set of conditions they can also be viewed as a description of the data graphs that do satisfy these conditions. Such descriptions may be used for a variety of purposes beside validation, including user interface building, code generation and data integration.

Security

We add to the the application security concept, where users/roles can be combined on the branches (paths), a new dimension, attribute grants. Infoton as a set of triples can't hide partial information, the process must be covered by the client. But we do not want to show everything to the user. With partial grants in roles, application filters the outgoing results.

Fuzzy Search

Approximate string matching (colloquially: fuzzy string searching) is implemented by default in search mechanism, supporting boolean querying by predicate to search constructuction.

Fuzzy search is the technique of finding strings that match a pattern approximately (rather than exactly). The problem of approximate string matching is typically divided into two sub-problems: finding approximate substring matches inside a given string and finding dictionary strings that match the pattern approximately. Fuzzy and exact search can be combined together in a search request.

Knowledge Representation

Large-scale knowledge representation (KR) with RDF unveils shortcomings which become obvious when facts have to be further qualified with contextual aspects (Infotons) to represent anything else but simplistic binary predicates. Motivated by requirements in different projects, in which we are required to store a large amount of information extracted from a heterogeneous set of documents and encoded in set of RDF triples as "data graphs"(Infotons), we adopt an architecture for modelling context in RDF knowledge bases. Our approach - based on well-researched theories of context in KR - avoids issues that other approaches face, by preserving standard RDF within a context and adding context semantics and relations between contexts around the standard.

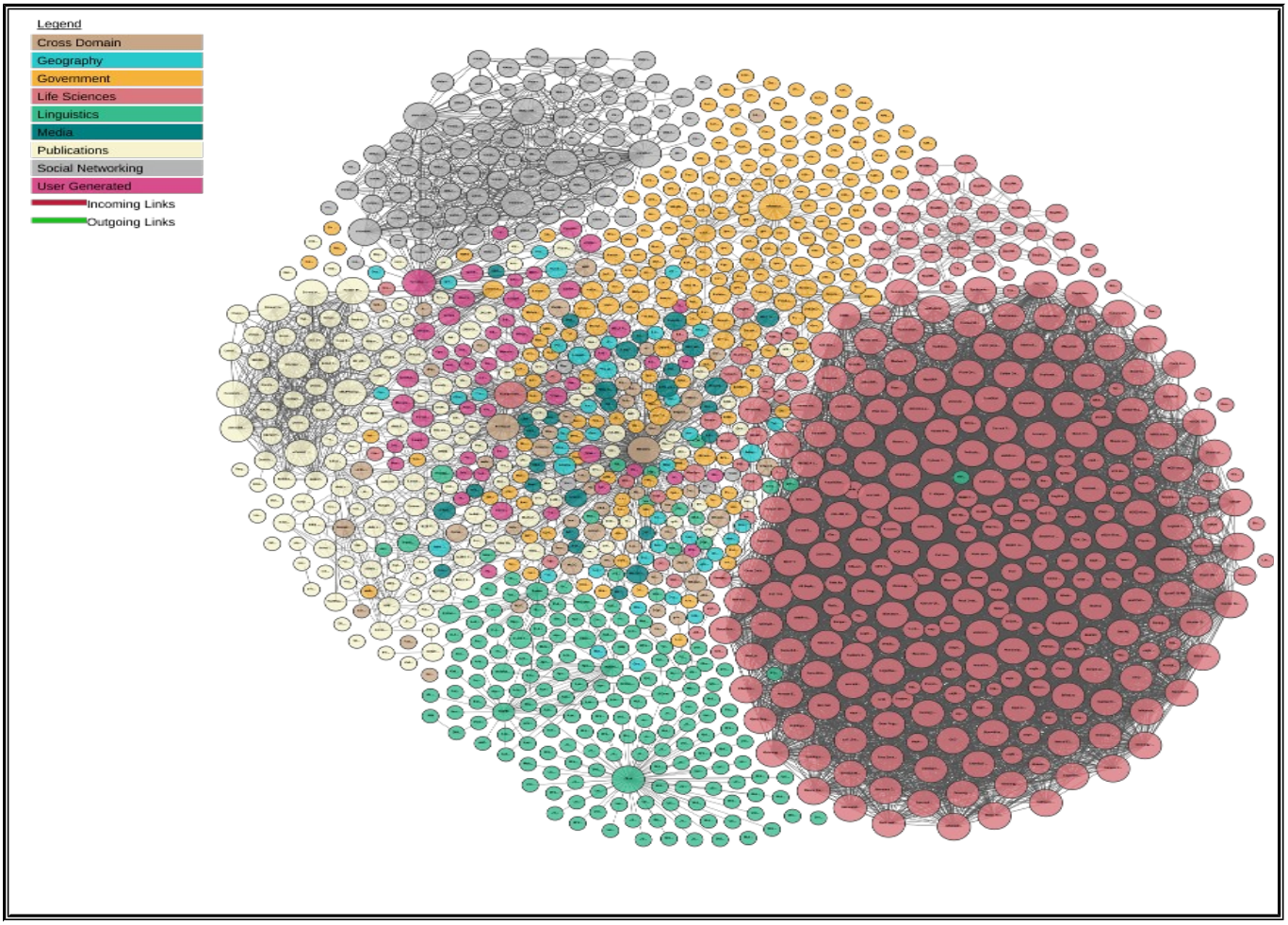

Fig.3 (source Wikipedia Linked Data)

REST Interface

Representational state transfer (REST) is a de-facto standard for a software architecture for interactive applications that typically use multiple Web services. In order to be used in a REST-based application, a Web Service needs to meet certain constraints; such a Web Service is called RESTful. A RESTful Web

service is required to provide an application access to its Web resources in a textual representation and support reading and modification of them with a stateless protocol and a predefined set of operations. By being RESTful, Web Services provide interoperability between the computer systems on the internet that provide these services. REST offers an alternative to, for instance, SOAP as method of access to a Web Service.

By using a stateless protocol and standard operations, RESTful systems aim for fast performance, reliability, and the ability to grow by reusing components that can be managed and updated without affecting the system as a whole, even while it is running.

Clients can use REST directly or onion-skin wrapper functions. We provide libraries for PostgreSQL, C/C++, JavaScript, R and Scala/Java.

Conclusions

For us, CBB application opens-up a lot of opportunities, specially in the case of handling large data sets. For instance: Weather Forecasting production system, where we have to combine a lot of external data sources, time-series (on-line feed of measurements of weather parameters, weather models, large GRIB structures (a set of n-dimensional matrix), radar data images, all in variety of formats, ontological description of meteorological and climate data collections, etc.) and the dissemination of the results, produced by our production system (approx. 75,000 products/day) and other background processes. At the end, it serves as data mart, and with an integration of reasoning, much more than just a data warehouse.

In fact, the solution is useful wherever you have something that resembles of graph data structure.

References

1. Gacitua, R., Mazon, J.N., Cravero, A.: Using Semantic Web technologies in the development of Data Warehouses: A Systematic Mapping (2019)
2. Bennett, D., Engelbrecht, J., Landau, D.:CM-Well: A Data Warehouse for Linked Data (2017)
3. Dayarathna, M., Herath, I., Dewmini, Y., Mettananda, G., Nandasiri, S., Jayasena, S., Suzumura, T.: Introducing acacia-rdf: An x10-based scalable distributed rdf graph database engine. In: 2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). pp. 1024{1032 (May 2016)
4. Peng, P., Zou, L., Ozsu, M.T., Chen, L., Zhao, D.: Processing sparql queries over distributed rdf graphs. The VLDB Journal 25(2), 243{268 (Apr 2016), https://doi.org/10.1007/s00778-015-0415-0
5. Punnoose, R., Crainiceanu, A., Rapp, D.: Sparql in the cloud using rya. Information Systems 48, 181 { 195 (2015), http://www.sciencedirect.com/science/article/pii/S0306437913000975
6. Um, J.H., Lee, S., Kim, T.H., Jeong, C.H., Song, S.K., Jung, H.: Distributed rdf store for effcient searching billions of triples based on hadoop. The Journal of Supercomputing 72(5), 1825{1840 (May 2016), https://doi.org/10.1007/s11227-016-1670-6

Addons: Reasoning and Multi Criteria Optimization

We are using new strategies and heuristics for solving multi-criteria optimization problems via ASP/SAT/SMT solvers (parallel processing), and not in a classic way by MILP/MIP/LP. In particular, the algorithms, based on conflict-driven learning (CDCL), allowing for non-uniform descents during optimization.

Knowledge representation in different context aims combinations of different processes merging together, like Job Shop Scheduling, shift-work optimization, selection of technology and machines, material, production conditions and energy consumption. Further, we address a small set of isolated solutions which can run together.

Graph/MultiGraph Fuzzy Search

As mentioned above, linked data are structured graph, so if we want to do something with, we have to enable calculus on. The graph isomorphism, subgraph isomorphism, and graph edit distance problems are combinatorial problems with many applications. Heuristic exact and approximate algorithms for each of these problems have been developed for different kinds of graphs: directed, undirected, labeled, etc. However, additional work is often needed to adapt such algorithms to different classes of graphs, for example to accommodate both labels and property annotations on nodes and edges.

We create Generalized Model to calculate the graph isomorphism, subgraph isomorphism, and graph edit distance problems, it is obvious that such a presentation can cover search of multi-graph/graph similarities. The model consists of labeled nodes and labeled directed/undirected edges, every node/edge has multiple properties, as key-value pairs, where the value can be single value or interval (like process tolerance, minimum and maximum, etc.). Certain nodes/edges have importance, so we have hard and soft constraints, and the minimization offers a set of answers.

Search a MultiGraph Similarities

A multi-graph is a set consisting of multiple graphs. Multi-graph similarity search aims to find the multi-graphs similar to the query multi-graphs from the multi-graph datasets. It plays important role in a wide range of application fields, such as finding similar drugs, searching similar molecule groups, similar subconstructions in large constructions, similar processes in knowledge base and so on.

MultiGraph Edit Distance

MultiGraph edit distance (GED) is a measure of similarity (or dissimilarity) between two graphs (NP-complete, formally APX-hard). A major application of graph edit distance is in inexact graph matching, such as error-tolerant pattern recognition in machine learning.

Brief description of the company

ACADEMA is an engineering software development company. In the early 90's, the company developed software for Civil Engineering, Mechanic of Structures and CNC Machining, but later on, in the mid of 90's, moved to Geographic Information Systems and Geostatistics. In the late of 90's, company started to develop ACADEMA Application Server, Java based software framework used to host the services and API to expose business logic and business processes for use by third-party applications.

The development is founding entirely from its own sources. Software engineering background is based on: Modeling of Processes, Numerical Analysis, Optimization Methods, Geometric Modeling (GIS Modeling), Topology and Formal Logic.

Practical results on the field

Few projects based on development on demand, built on ACADEMA Application Server (licensed).

- Weather Forecast Production System (Environmental Agency of the Republic of Slovenia) 19th EGOWS meeting programme: https://www.arso.gov.si/egows2008/program.html, results of production system https://meteo.arso.gov.si/met/en/app/webmet
- Mass Appraisal of Real Property Modeling System (Surveying and Mapping Authority of the Republic of Slovenia)
- GeaBios - Public Service - GIS and Astronomy
  https://www.geabios.com
  https://en.wikipedia.org/wiki/GeaBios
- More information on subject https://www.academa.si, https://en.wikipedia.org/wiki/Academa