# Prilagodba modela jezera podataka za pohranu i obradu vremensko-prostornih tokova podataka

**Hlupić, Tomislav**

Sveučilište u Zagrebu
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Tomislav Hlupić

# PRILAGODBA MODELA JEZERA PODATAKA ZA POHRANU I OBRADU VREMENSKO-PROSTORNIH TOKOVA PODATAKA

DOKTORSKI RAD

Zagreb, 2022.

Sveučilište u Zagrebu
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

TOMISLAV HLUPIĆ

# PRILAGODBA MODELA JEZERA PODATAKA ZA POHRANU I OBRADU VREMENSKO-PROSTORNIH TOKOVA PODATAKA

DOKTORSKI RAD

Mentor: prof. dr. sc. Mirta Baranović

Zagreb, 2022.

University of Zagreb
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Tomislav Hlupić

# ADJUSTMENT OF THE DATA LAKE MODEL FOR STORING AND PROCESSING OF SPATIO-TEMPORAL DATA STREAMS

DOCTORAL THESIS

Supervisor: Professor Mirta Baranović, PhD

Zagreb, 2022.

Doktorski rad izrađen je na Sveučilištu u Zagrebu

Fakultetu elektrotehnike i računarstva

na Zavodu za primjenjeno računarstvo.

Mentor: prof. dr. sc. Mirta Baranović

Doktorski rad ima 156 stranica

Doktorski rad br.:

# O mentoru

**Mirta Baranović** rođena je u Zagrebu 1952. godine. Diplomirala je, magistrirala i doktorirala u polju računarstva na Sveučilištu u Zagrebu Fakultetu elektrotehnike i računarstva (FER), 1976., 1983. odnosno 1997. godine.

Od kolovoza 1976. godine radi na ETF-u, kasnijem FER-u, na Zavodu za primijenjenu matematiku, kasnijem Zavodu za primijenjeno računarstvo. Od 2010. do 2014. godine bila je predstojnica Zavoda za primijenjeno računarstvo. U zvanje redovite profesorice u trajnom zvanju izabrana je 2018. godine. Izvodila je nastavu iz predmeta vezanih uz baze podataka, na preddiplomskom, diplomskom i poslijediplomskom studiju te je kao mentor vodila 7 doktorata, 19 magistarskih radova te oko 300 diplomskih i završnih radova. Sudjelovala je na tri i vodila je dva znanstvena projekta financirana od Ministarstva znanosti i obrazovanja Trenutno sudjeluje na projektu: Napredne metode i tehnologije u znanosti o podatcima i kooperativnim sustavima (DATACROSS) - Istraživanje u znanosti o podatcima. Objavila je više od 70 radova u časopisima i zbornicima međunarodnih konferencija u području baza podataka, skladišta podataka i poslovne inteligencije, semantičkog weba i geoprostornih baza i tokova podataka.

Prof. Baranović članica je stručnih udruga IEEE, ACM, MIPRO i CIGRE. Sudjeluje u međunarodnom programskom odboru znanstvene konferencije te sudjeluje kao recenzentica u većem broju konferencija i časopisa. Od 2003. do 2005. godine bila je Počasna predsjednica European Federation of National Maintenance Societies (EFNMS), a od 2005 do 2009. članica Upravnog odbora EFNMS-a. Godine 1997. nagrađena je srebrnom plaketom "Josip Lončar" FER-a za posebno istaknutu doktorsku disertaciju. 2010. godine dodijeljeno joj je Posebno priznanje Rektora Sveučilišta u Zagrebu za izuzetan doprinos u provedbi i promociji međunarodne suradnje, 2012. godine nagrada IBM – International Informix Users Group Award. Godine 2014. nagrađena je zlatnom plaketom FER-a „Josip Lončar“ te 2015. godine nagradom Hrvatske sekcije IEEE za izniman doprinos u inženjerskom obrazovanju.

## About the Supervisor

**Mirta Baranović** was born in Zagreb in 1952. She received her B.Sc., M.Sc. and Ph.D. degrees in computer science from the University of Zagreb, Faculty of Electrical Engineering and Computing (FER), Zagreb, Croatia, in 1976, 1983 and 1997, respectively.

Since August 1976 she has been working at the ETF, later the Faculty of Electrical Engineering and Computing, Department of Applied Computing. From 2010 to 2014, she was the head of the Department of Applied Computing. She was elected a full professor in Computer Science in 2018. She has taught database-related subjects at the bachelor, graduate and post-graduate level, and she mentored 7 PhD theses, and about 300 graduation and final theses. She has participated in three and led two scientific projects funded by the Ministry of Science and Education. Currently, she is participating in the project: Advanced Methods and Technologies in Data Science and Cooperative Systems (DATACROSS) - Research in Data Science. She has published more than 70 papers in journals and proceedings of international conferences in the field of databases, data warehouses and business intelligence, semantic web and geospatial databases and data streams.

Prof. Baranović is member of the professional associations IEEE, ACM, MIPRO and CIGRE. She participates in the international program committee of scientific conferences and participates as a reviewer in several conferences and journals. From 2003 to 2005 she was Honorary President of the European Federation of National Maintenance Societies (EFNMS), and from 2005 to 2009 she was a member of the Board of Directors of EFNMS. In 1997 she received silver medal "Josip Lončar" from FER for outstanding Ph.D. thesis. In 2010 she was awarded the Special Recognition of the Rector of the University of Zagreb for outstanding contribution to the international cooperation, and in 2012 she received the IBM - International Informix Users Group Award. In 2014, she received gold medal "Josip Lončar" from FER and in 2015 the Croatian IEEE Section Award for Outstanding Contribution to Engineering Education.

# Zahvala

Zahvaljujem svojoj mentorici prof. dr. sc. Mirti Baranović na svim savjetima, pomoći i potpori u istraživanjima i izradi doktorskog rada.

Hvala mojim roditeljima na nesebičnoj potpori tijekom cijelog obrazovanja.

Hvala mojoj supruzi Sari na svim riječima potpore, strpljenju, pomoći i neograničenom razumijevanju.

# Sažetak

Jezera podataka predstavljaju moderni pristup pohrani i analizi velikih podataka, koji u njih pristižu u strukturiranom, nestrukturiranom i polustrukturiranom obliku velikom brzinom i u velikim količinama, a veliku prednost, zahvaljujući skalabilnosti nalaze u potpori obradi i pohrani tokova podataka. Među njima ističu se prostorno-vremenski tokovi podataka, čiji su izvori često uređaji koje objedinjeno nazivamo Internet stvari. Ova vrsta tokova podataka ima naglašenu geoprostornu komponentu, koja omogućuje gotovo stvarnovremensko praćenje kretanja izvora podataka u prostoru kroz vrijeme.

Prilagodba jezera podataka za potporu prostorno-vremenskim podacima obavlja se na razini pohrane podataka i obrade podataka. Doprinos ovog rada sastoji se od prijedloga modela unificiranog sloja pristupa podacima koji u sebi sadrži potporu za prostorno-vremenske tokove podataka. Uz njih, predloženi su algoritmi potpore operacijama nad tipovima prostorno-vremenskih tokova podataka i ugrađeni u unificirani sloj pristupa podacima.

Uz unificirani sloj pristupa podacima, predložen je model repozitorija metapodataka koji ima ulogu potpore izgradnji podatkovnih cjevovoda unutar unificiranog sloja pristupa podacima i procesima upravljanja podacima. U repozitoriju metapodataka podaci su opisani do razine pojedinog atributa u obrađenom entitetu.

Predloženi modeli verificirani su kroz obradu i pohranu heterogeno strukturiranih prostorno-vremenskih tokova podataka u cijelom jezeru podataka, zaključno s njihovom vizualizacijom. Rezultati verifikacija pokazali su kako predloženi unificirani sloj pristupa podacima omogućuje dinamičku izgradnju podatkovnih cjevovoda s potporom za prostorno-vremenske tokove podataka.



**Ključni pojmovi:** jezera podataka, prostorno-vremenski tokovi podataka, modeli jezera podataka, unificirani sloj pristupa podacima

# Adjustment of the Data Lake model for storing and processing of spatio-temporal data streams

Data Lakes have risen as a Big Data analytics and storage foundation, mostly as a response to the exponential increase in data volume and velocity, which became an insurmountable challenge for traditional analytics solutions based on relational databases. Data Lakes act as a central repository for efficient storing and processing of Big Data in near real-time, with the basis being distributed systems which, due to their features, provide the support for scalability and parallelism. Moreover, as the data evolved from mostly structured, to semi-structured, towards mostly unstructured, the analytics solutions had to be adapted to the rising trends. The Data Lakes, following the philosophy of storing the data in their raw, unprocessed format, contributed to the processing overhead decrease, thus allowing near-real-time data processing analytics. This made Data Lakes, as a versatile data analytics platform, a choice for data-driven organizations looking to gain insights from the Big Data collected and stored in them.

As the Data Lakes have technically evolved towards near real-time processing and analytics solutions for heterogeneous data, the evolution of their formal definition and models did not follow the same pace. This resulted in a certain discrepancy between various suggested models, each having a focus on a different areas of definitions and models and, in the end, not having an unadjusted approach towards Data Lake architectures. This led to many organizations struggling to leverage their implemented Data Lakes to achieve the full potential of the stored data.

Spatio-temporal data streams have become widely prevalent as data sources over time. This is the result of the growth of the devices commonly referred to as the Internet of Things (IoT), especially the moving sensors that are characterized by providing both geolocation and time component in their generated data. The existing formal frameworks that describe spatio-temporal data types and their operations were not adapted to Big Data sources, especially the ones having their data stored in the unstructured *blob* data type.

The contributions of this thesis are as follows:

1) A proposal of a unified data access layer for Data Lakes that support spatio-temporal data types

2) An enhancement proposal of the Data Lake model with data types for spatio-temporal data streams through a unified data access layer
3) Proposal of algorithms for supporting operations on spatio-temporal data types and their integration into the unified data access layer of Data Lake

Data Lakes were initially based on the horizontally and vertically scalable distributed analytical systems that could handle storing and processing of unstructured data, as the Big Data came from heterogeneous data sources that generate high-volume, high-speed, and high-velocity data. These aforementioned features, commonly abbreviated as 3V, directly affected the necessity of system scalability, as those features can vary over time. As mentioned previously, unlike Data Lakes, traditional data warehouses which were previously the primary analytics solutions, do not have the necessary scalability capabilities (at least not to the full extent) to handle Big Data.

Data lakes follow the paradigm of extract-load-transform (ELT), or its extended form extract-load-transform-analysis (ELTA). Unlike data warehouses, Data Lakes rely on distributed systems that allow fast data retrieval and transformation through parallel processing tasks. As the data warehouses use the extract-transform-load (ETL) paradigm, the integration process usually becomes a bottleneck that prevents their near real-time capabilities. ELT paradigm is closer to the data ingestion process, which is a subpart of integration that does not put transformation, as the most intensive phase, prior to the data storage. Additionally, increasing the volume or rate of incoming data does not necessarily degrade real-time analytics support, but can be addressed by scaling the entire system.

The data pipelines, used in Data Lakes, therefore do not end with the data being stored. It is important to note that within data pipelines, processes do not stop with data storage. On the contrary, after the data storage process, it is possible to continue using these data in the following defined processes, which link complex data processing into a unique sequence.

Data Lakes were originally defined with a two-zoned architecture model, with a landing zone and a raw data zone. Therefore, their initial definition could be summed as:

*Data Lake is defined as a heterogeneous data repository with data stored in their raw, unmodified format.*

The two-zoned architecture provides Data Lakes with their basic features: raw data storage and scalable data access to the stored data. However, as the Data Lakes usage became more

prominent, it became clear the existing architecture models have to be adapted into the ones oriented towards business support.

Therefore, the initial definition is adapted into the following:

*Data Lake is defined as a heterogeneous data repository with data stored in their raw, unmodified format and other formats adjusted for further usage.*

Using this definition, the Data Lakes are formally enabled to step away from the two-zone architecture into multi-zone and multi-layered architectures. Through the research of the existing literature in the Data Lake architecture domain, several prominent models were analyzed and compared: Imnon's data pond model, layered Data Lake model, Zaloni data model based on zones and its derivative Data Vault based Data Lake model, Lambda architecture and Data Lakehouse.

Imnon's model, through analysis, was deemed to have a significant shortcoming of not being able to reprocess stored data due to a lack of raw data zone storage (the data could not change the storage type), but it was the first one to formally split hot and cold storage zones. The layered Data Lake model, besides storage, also defines the data pipelines, metadata storage and extraction, and layers of data access. Zonal Data Lake models, on the other hand, define various zones with data stored in formats and models adjusted for further analysis and application access. With Lambda architecture, the data processing was defined by splitting batch and fast (real-time) data in order to apply analysis over the complete dataset. Data Lakehouse model combined approaches of using Data Lake for storing raw data and Data Warehouse for processed data adjusted to a well-known structured analytical format.

In order to determine the types of data sources and types of data streams as their subsets, their systematization was needed. Two levels of systematization were done: one on the data source level and one on the data streams level.

Structured data are data with a known structure, which is defined in advance. This means that the data itself and their structure are known before the data appear in a system. Structured data can be delivered through interfaces that support structured data, and the delivery is processed using embedded procedures or ETL processes. Structured data can be mapped to semi-structured through automated metadata-based processes, and operators can be applied to structured data to support the processing of spatio-temporal data streams.

Semi-structured data have a given structure that is not as strict as in the case of structured data. Semi-structured data can be stored in their raw form through data ingestion processes, but it can also be included in data pipelines that perform data integration processes using object-relational mapping (ORM) procedures. These processes can result in the creation of new documents or flattened tuples in 1NF, while it is also possible to normalize the flattened tuples to 3NF using advanced methods based on metadata. Metadata for semi-structured data include information such as the data sources, entities, attributes, and mapping functions.

Unstructured data are data without a fixed structure at the time of delivery, but which is acquired after it is received or stored in persistent memory. Unstructured data sources are minimally described with metadata about the source itself and the entities within the source. Mapping from unstructured data into semi-structured or structured data is done through dedicated mapping functions, and it is easier to be made to semi-structured data than to structured data. Moreover, unstructured data can be integrated with semi-structured and structured data in data pipelines using appropriate processes.

Data streams, as a subset of data sources, are special types of data sources characterized by the continuous delivery of time-variant data. They are similarly divided into structured, semi-structured, and unstructured data streams. Structured data streams contain all supported data types, have a previously known structure with explicitly or implicitly defined time dimension, and connect to structured data interfaces that support the CQL query language and its associated operators, predominantly stream-to-relation operators. Semi-structured are easier to connect to heterogeneous schemes within data pipelines than structured data streams and can be concatenated to create a new schema. When working with semi-structured data streams, stream-to-document operators are mainly used within data ingestion processes. Unstructured data flows do not have a previously defined structure or their structure does not correspond to relational or semi-structured schemas. When dealing with unstructured data, its features have to be extracted from metadata in order to define the temporal and spatial domain of the records.

Regarding the spatio-temporal data storage in Data Lakes, the structured data are stored as tuples in post-relational databases and require support for spatio-temporal data types and operations. Unstructured spatio-temporal data are stored as blob, which in case of HDFS usage can be exploited using systems such as SpatialHadoop and ST-Hadoop. Semi-structured spatio-temporal data, on the other hand, are stored in document and graph databases.

Research in the Data Lake domain included research of existing definitions, analysis of known Data Lake models, and definition of the Data Lake feature set that includes aspects such as data quality, access control and restriction, data organization, data types, and scalability. Further research in this domain focused on data management in Data Lakes, defining areas of knowledge, and Lamba and Kappa architecture. Research in the domain of metadata in Data Lakes included exploring different types of metadata such as structural, inherent, semantic, technical, operational, and business metadata, as well as metadata classes such as intraobject, interobject, and global metadata. Additionally, the use of metadata in edge computing and Data Lake integration with edge computing components was also analyzed.

The research in the domain of semi-structured data focused on formally defining the semi-structured data as the attribute-value pairs list (AVPL), defined as $(a \in A) \wedge (v \in V) \rightarrow \{(a, v)\} \in D$. In that definition, an attribute represents an ordered set of one or more variables, and a value is an instance or set of instances that are associated with the corresponding attribute and its variables. An object of AVPL type can be a component of another AVPL object. Moreover, the schema construction operations are defined, as well as the definition of schema construction by using a separate set of operators from one or more AVPL. Finally, the semi-structured schema operators are defined as follows: creation of a class by instance collection, object merging, class merging, object composition, class composition, and inclusion operators.

In the domain of spatio-temporal data types and operations, the focus was to cover the existing research and frameworks. Spatio-temporal data types and operations include base, spatial, and time data types, as well as range and temporal types. They also include relational types identifier, tuple, and relations, as well as stuple and stream types. Operators on these data types include stream-to-relation, relation-to-stream, and relation-to-relation, as well as sliding window operators. The analyzed formal framework for working with geospatial data streams is based on many-sorted algebra and second-order signatures, using the aforementioned data types and their operators. In this framework, a moving point is observed in three or four dimensions with mapping from a continuous time to a continuous spatial domain.

The initial step in adjusting the Data Lake model for spatio-temporal data streams was extending the existing set of data types and operations, mentioned in the previous paragraph. Data types set is extended by adding a *blob* (Binary Large Object) data type in order to support unstructured data. The *blob* data type was defined in the SQL:1999 standard, but its full impact was seen with the rise of unstructured data. Even though most of the existing database systems

technically supported using it, Data Lakes are largely dependent on data stored as *blob*, which made it an indispensable part of the data types set.

Moreover, it was identified that the basis for the adjusted Data Lake model should be semi-structured data in order to support mappings between unstructured and semistructured, as well as mappings between semi-structured and structured data. Along with that, the operators supporting the usage of semi-structured data based on defined AVPL and the stream-related operators that support the usage of semi-structured data streams were defined. There are two sets of operators, the first one being operators of type document-to-document (selection, projection and join operators), similar to relation-to-relation operators, and the second one being stream-to-document operators. The second set of operators is closely related to existing sliding window operators defined for structured data streams. This set consists of operators NOW, LAST, UNBOUNDED, PAST, SINCE, and RANGE, together with the definition of spatial windows and partitioned windows based on AVPL.

The proposed Data Lake model is divided into two layers: the data processing layer and the data storage layer. The data processing layer has two sets of interfaces: external and internal. The external interfaces are responsible for collecting incoming data and transforming them into a form that can be accessed by external systems. The internal interfaces, on the other hand, are responsible for exposing data to the data storage systems and accepting data from them.

Moreover, the proposed model integrates two main approaches for Data Lake architecture, zonal and layered. By their integration, the proposed model provides a two-dimensional view of the Data Lake architecture, giving them equal importance.

The data storage layer, on the other hand, has interfaces that communicate with data storage systems and support the storage of structured, semi-structured, and unstructured data. Each form of data storage requires a separate interface, and data is sent to and received from these interfaces in a form that is adjusted for a particular interface. This is the responsibility of the data access layer, which uses its internal interfaces to expose data in the right form for each data storage system.

In addition to the data processing and data storage layers, the proposed model also defines Data Lake zones, which are defined by metadata and represent a logical division of the Data Lake based on the organization of data extraction, storage, and presentation. The zones themselves affect the tasks and complexity of individual data pipelines, and they are designed to allow data to be consumed in different forms depending on the needs of a particular zone.

Overall, the proposed data lake model is presented to external systems as a single unit, with all supported data types and operations specified in a uniform way. This provides a solid foundation for unified data access and processing, and to a certain extent, it limits the technological solutions that can be used to implement the Data Lake.

The proposed Data Lake model is divided into six zones with a specific purpose and function within it. These zones are defined as data landing zone, raw data zone, adjusted data zone, adjusted access zone, research zone, and application access zone. The data landing zone is where data are first received from external systems via appropriate interfaces. They are temporarily stored in this zone until their processing and moving into the raw data zone. The raw data zone contains persistently stored data in their original, unaltered form. The adjusted data zone contains data that have been adjusted through processes of integration or refinement in data pipelines, while the adjusted access zone is built on top of the adjusted data zone but introduces an additional layer of access level control. The research zone contains all data available for user analysis, and the application access zone contains data that are adapted for application usage.

The data processing layer in a Data Lake is responsible for collecting, processing, and presenting structured, semi-structured, and unstructured data in a unified way. It consists of two main components: a metadata repository and a unified data access layer. The metadata repository enables high-quality data monitoring, while the unified data access layer adapts the form of incoming or outgoing data and directs it to data storage interfaces. The data processing layer is implemented through a series of data pipelines, while communication with external systems is achieved through access interfaces. The data processing layer also serves as a virtualization layer, hiding the complexity of the Data Lake model and enabling data from multiple storage systems to be presented in a single structure.

The data storage layer of the Data Lake model consists of three storage system groups that support the storage of structured, unstructured, and semi-structured data. Data are routed to the appropriate storage system through the data processing layer based on the data in the metadata repository. This enables the storage of data in its original form, as well as its replication and adaptation for storage in other forms. The storage of the same data in different forms is particularly important in the research zone, where access to all data in the Data Lake is available. The use of NoSQL databases, such as document and graph databases, enables the storage and analysis of semi-structured data.

The Unified Data Access Layer (UDAL) is the middleware in the Data Lake model. Its tasks include collecting data from external sources using interfaces, creating data pipelines for transferring data between interfaces, and exposing data to data storage systems. The UDAL is based on data pipelines, which consist of at least one input $\mathcal{S}_u$ and one output interface ($\mathcal{S}_i$), as well as a manager ($\mathcal{M}$) with an associated procedure ($\mathcal{p}$). Data pipelines that work with individual data structures form the UDAL. Input and output interfaces enable data to be entered into the data pipeline and exposed by the data storage layer, respectively.

The real-time interface manager in UDAL is responsible for communication between the input interfaces where data is collected and other managers within the data pipelines. It uses polling to periodically query the input interfaces for new data. There are two types of communication: the push method, where the data source communicates directly with the input interface, and the pull method, where the manager queries the input interface for new data. In both cases, the real-time interface manager passes the data into the data pipeline to begin the data ingestion process.

Other managers within the data pipeline include the stream-to-batch mapping manager, the data structure mapping manager, and the data merging manager. These managers are responsible for converting data streams to data series, applying mapping operators to records, integrating and merging structured and semi-structured data.

The metadata repository manager is responsible for maintaining communication between the metadata repository and the managers in the UDAL. This allows for central storage and retrieval of metadata about data structures and processes within the Data Lake. The subscription manager manages subscriptions to data and controls the correctness of data delivery to output interfaces.

The temporary raw data container manager provides temporary storage for data in the fast data layer of the Lambda architecture before it is written to persistent memory. This ensures its availability through specialized output interfaces until it is processed and written to persistent memory. The manager also takes care of removing data from the temporary container when necessary.

The metadata repository is an important part of the data lake architecture. It enables the tracking of data structure and evolution, as well as unified data processing. The metadata repository is divided into structural, characteristic, and semantic metadata, as well as intra-object and inter-object metadata. Additionally, the metadata repository can be used as a basis

for building a system for data processing and storage. It can be extended with additional features and classes to support data management processes. This makes the metadata repository a central place for storing and describing data processing, enabling dynamic data management. The metadata repository defines three groups of classes: intraobject data classes that define a particular object and associated attributes, interobject data classes that specify relationships between data sets and attributes, and data processing classes that specify the monitoring of data processing on the batch granularity.

The unified data access layer (UDAL) contains algorithms that are related to merging data streams in different formats, such as merging AVPL with AVPL streams and merging AVPL streams with relational data streams. These algorithms support UDAL managers in performing tasks such as mapping records from the tuple to AVPL, mapping blob files to AVPL, and managing records in the temporary container. Additionally, the UDAL includes a refresh algorithm for the real-time interface manager. These algorithms enable the UDAL to effectively process and manage incoming data.

In this review of advanced analysis tools, a qualitative analysis was performed on tools that are currently among the most widely used or significant in their respective domains. The analysis was divided into several categories based on their features, and for each category, the ability to process spatial and spatio-temporal analyzes was examined. The categories that were analyzed are general-purpose data science tools, framework extensions, extensions of programming languages with libraries, extensions within the Big Data ecosystem, and visualization tools in the context of spatio-temporal data.

The prototype of the data lake model was developed using the Docker container platform. The use of containers allows for interoperability between different systems and independence from infrastructure. The data lake prototype includes several containers that are used to implement the data processing and data storage layers. The data storage layer contains containers for structured, semi-structured, and unstructured data. The data processing layer contains containers with applications and interfaces, as well as the metadata repository, including containers for Apache Kafka, Apache Zookeeper, and Apache Spark, among others, which form the UDAL. The prototype also includes containers for the real-time interface manager, the subscription manager, and the temporary container manager. Moreover, UDAL implementation includes support for spatio-temporal data and includes algorithms for merging data streams and mapping data structures. The metadata repository is implemented as a post-

relational database in a SQL Server container. Overall, the use of containers in the prototype enables easy expansion and integration with additional systems and applications.

The Data Lake models have evolved from two-zone architectures to more complex multi-zone architectures with adjusted data, with mandatory support for analytics and visualization of semi-structured and unstructured data. Data processing in the proposed Data Lake model is performed through a specialized middleware name Unified Data Access Layer (UDAL that has several tasks, including data integration and ingestion, retrieval and storage of data in the data storage layer, support for external applications, and support for near real-time analysis. UDAL is based on data pipelines and managers, and the metadata repository is mandatory in the data processing layer. Future research includes the automatic extraction of metadata from raw data and its incorporation into UDAL, integration of the system for entity matching and merging into UDAL, and the development of custom Data Lake models depending on their deployment type.

# SADRŽAJ

# Uvod

U proteklih 15 godina, ponajviše zahvaljujući Web 2.0 tehnologijama, količina dostupnih podataka eksponencijalno je narasla u odnosu na početak stoljeća. Ujedno je porasla i količina izvora podataka, od mobilnih uređaja, preko nosivih uređaja do sofisticiranih sustava senzora kakvi se, primjerice, ugrađuju u samovozeća vozila. Takvi izvori informacija, uz podatke generirane kroz društvene mreže, stvorili su potrebu za brzom pohranom i analizom velike količine heterogenih podataka koji stvarnovremenski pristižu u analitičke sustave.

Kao odgovor na takve zahtjeve prvo su razvijeni raspodijeljeni sustavi za pohranu velikih količina nestrukturiranih podataka, što je postao temelj jezera podataka. Postojeći sustavi za rad s podacima, gdje su dominirali relacijski sustavi za upravljanje bazama podataka (RSUBP), ne podržavaju skaliranje zahtjeva u dovoljnoj mjeri. Jezera podataka, temeljena na raspodijeljenim sustavima, kroz horizontalno i vertikalno skaliranje, nametnula su se kao jedno od rješenja za rad s velikim podacima.

Velika brzina industrijskog razvoja u domeni jezera podataka stvorila je diskrepanciju između funkcionalnosti, implementacije i korištenja samih jezera podataka. Sama jezera podataka dugo su bila izjednačavana s Hadoop ekosustavom, što je dovelo do inicijalnog shvaćanja jezera podataka kao repozitorija nepromijenjenih podataka. Kako bi se podržali novi zahtjevi, razvijan je ekosustav alata i sustava za rad s podacima koji su djelomično rješavali detektirane nedostatke u implementacijama. Pritom je često zanemarivan proces upravljanja podacima, što je ponekad rezultiralo stvaranjem močvare podataka (eng. *data swamp*) i posljedičnim prestankom korištenja sustava.

Jezera podataka, u svojoj osnovi, nemaju formalno definiranu strukturu niti način zapisa podataka. Iz tog razloga, modeli jezera podataka u startu su definirani iznimno jednostavno – podaci su pohranjivani u datotečni sustav u nepromijenjenom obliku. Time je omogućena analiza podataka kroz definiranje sheme u ovisnosti o samom zahtjevu, odnosno sheme na upit (eng. *schema-on-read*). U takvom načinu rada, sustav se oslanja na paralelizam i skalabilnost kako bi se podaci analizirali u gotovo stvarnom vremenu.

Budući da većina sustava u praksi ima određenu granicu paralelizma, značajno opterećenje kroz veliki istovremeni broj upita nad jezerom podataka može rezultirati smanjenim performansama. Jedno od mogućih rješenja rasterećenja sustava je definiranje poznate sheme

nad podacima koja bi korisnicima i aplikacijama omogućila pristup podacima bez dodatnog troška stvaranja sheme u danom trenutku.

Alati za analizu, bilo kroz upite ili kroz vizualizacije, u početku su se oslanjali na relacije i relacijske baze podataka, odnosno strukturirane podatke. Kroz intenzivno korištenje polustrukturiranih zapisa podataka kao što su JSON i XML u modernim oblicima komunikacije, potpora za polustrukturirane podatke u alatima se povećavala. Na taj način, izvori podataka proširili su i na zapise stvorene kao dnevnički zapisi sustava ili rezultate određenih akcija u web aplikacijama. Web aplikacije u velikom broju slučajeva koriste JSON oblik zapisa u svojem radu, što je rezultat uključivanja programskih okvira temeljenih na programskom jeziku JavaScript u izradu korisničkih sučelja.

Polustrukturirani podaci pogodniji su za potporu sustavima u kojima se očekuje češća promjena strukture podataka od relacijskih baza podataka jer nude veću fleksibilnost u definiciji sheme. Pritom je ključna činjenica da je shema poznata, odnosno poznat je maksimalni skup atributa u shemi, a dolaskom novih atributa shema se jednostavno proširuje bez potrebe za eksplicitnim operacijama. Dodatna prednost polustrukturiranih podataka u odnosu na strukturirane je jednostavna potpora ugniježđenim i hijerarhijskim podacima bez potrebe za normalizacijom podataka.

Uz poboljšanja sustava za pohranu polustrukturiranih podataka, ali i zahvaljujući njihovim odlikama, prvenstveno skalabilnosti, korištenje polustrukturiranih podataka u analitičke svrhe postalo je u jednakoj mjeri prihvatljivo kao i korištenje postrelacijskih baza podataka. Pored toga, kada izvori podataka već dostavljaju zapise u polustrukturiranom obliku, smanjuje se količina troška potrebnog za njihovu dodatnu obradu i pohranu. Uzevši u obzir kako broj polustrukturiranih izvora podataka raste brže od strukturiranih, u raspodijeljenim sustavima polustrukturirani su zapisi češće prisutni.

Kao posljedica razvoja sustava koji počivaju na korisničkim interakcijama i stvaranju sadržaja, uz polustrukturirane podatke eksponencijalno se povećao i volumen nestrukturiranih podataka. Oni se protežu od multimedijskih zapisa, tekstualnih zapisa, kombiniranih podataka o korištenju aplikacija sustava do podataka bez određene strukture, nastalih u uređajima. Upravo je ovaj segment podataka, koji se odlikuju velikim volumenom, brzinom i heterogenošću, najviše utjecao na potrebu razvoja jezera podataka.

Nestrukturirani podaci također u značajnoj mjeri dolaze s mobilnih uređaja koji, uz korisnički generirane podatke, stvaraju prostorno-vremenski ovisne podatke ili njima proširuju

ostale podatke. U takve podatke ubrajaju se dnevnički zapisi mobilnih aplikacija i operacijskog sustava, s različitim konačnim svrhama korištenja.

## Motivacija

Ubrzani razvoj analitičkih sustava na kojima se temelje jezera podataka ujedno je donio i dugoročni izostanak formalnih definicija i modela takvih sustava. Tako se modelom jezera podataka dugo smatrala osnovna arhitektura sa zonom prihvata podataka i zonom upisanih nepromijenjenih podataka. Evolucijom jezera podataka pojavila se potreba za formalnom arhitekturom, pri čemu su prisutna tri glavna modela, predložena u radovima [40], [41] i [42]. Kao dio cjelovitog sustava, proces obrade podataka također nije u većoj mjeri formalno definiran, budući da je obrada podataka u jezerima podataka dugo bila izjednačena s procesom gutanja podataka.

Tokovi podataka, kroz razvoj tehnologije i telekomunikacijske potpore, s vremenom počeli su povećavati svoj volumen i brzinu dolaska podataka. Za razliku od tradicionalnih analitičkih sustava, koji se ne mogu jednostavno skalirati, jezera podataka mogu se nositi s takvim povećanjima zahtjeva. To je svakako jedan od glavnih razloga zašto su jezera podataka, u pogledu potpore tokovima podataka, zasigurno bolji izbor analitičkog sustava od skladišta podataka temeljenih na relacijskim SUBP.

Prostorno-vremenski tokovi podataka predstavljaju posebnu skupinu tokova podataka, budući da u sebi nose podatke u prostornoj i vremenskoj domeni. Dok je vremenska domena prisutna u svim tokovima podataka, bilo eksplicitno ili implicitno, prostorna domena podataka zahtijeva prilagodbu modela jezera podataka. Takva se prilagodba odnosi na definiciju skupa podržanih tipova podataka i njihovih povezanih operacija, ali i na model jezera podataka. Prilagođeni model jezera podataka mora, pored pohrane nepromijenjenih podataka, omogućiti analizu podataka u prostornoj i vremenskoj domeni, kao i način prihvata podataka iz tokova podataka i njihovu obradu kroz podatkovne cjevovode.

Količina izvora prostorno-vremenskih tokova podataka konstantno se povećava te oni postaju sve kompleksniji u strukturama podataka. Iz tog razloga, prijelaz iz strukturiranih podataka u prostorno-vremenskim tokovima podataka u polustrukturirane i nestrukturirane došao je prirodno. Strukturirani podaci povećavanjem kompleksnosti podataka i sami rastu u kompleksnosti modela, što zahtijeva dorade ne samo u izvorima podataka već i u podatkovnim cjevovodima sve do perzistentne pohrane. Kod polustrukturiranih i nestrukturiranih zapisa, s

druge strane, povećanje kompleksnosti ili promjena strukture podataka ne zahtijeva nikakve eksplicitne promjene, kao što je slučaj kod strukturiranih podataka.

Iako s tehničke strane dorade u navedenim slučajevima nisu potrebne, one ipak utječu na dio procesa u jezerima podataka, poglavito na procese upravljanja podacima. Sve promjene strukture trebaju biti registrirane kroz promjene u metapodacima, za što je zadužen repozitorij metapodataka. U njemu svaka promjena strukture rezultira dodavanjem zapisa o promjenama u različitim klasama metapodataka. Primjerice, dodavanje atributa u zapis u strukturu podataka unutar repozitorija podataka praćeno je dodavanjem novog zapisa u klasu atributa te pripadnim zapisima o evoluciji samog zapisa, ovisno o definiciji samog repozitorija. Na ovako jednostavnom primjeru već je jasna nužnost uspostave kvalitetnog repozitorija metapodataka unutar samog jezera podataka.

U odnosu na postojeće definirane skupove podržanih prostorno-vremenskih tipova podataka, u svrhu potpore radu s nestrukturiranim podacima nužna je njihova prilagodba. Jezera podataka, u odnosu na relacijske SUBP, uvelike se oslanjaju na pohranu i rad s nestrukturiranim tipovima podataka. Budući da nestrukturirani podaci svoju konačnu strukturu dobivaju tek prilikom analize, potreban je tip podataka koji će ih uniformno predstavljati u sustavima tipovima podataka.

Tip podataka *blob* u potpunosti opisuje nestrukturirane podatke. Kako su oni varijabilne veličine i varijabilnog sadržaja, ne mogu se jedinstveno iskazati nijednim od dosad definiranih tipova podataka. Iz tog razloga, proširenje sustava tipova podataka novim tipom podatka *blob* nužno je za prilagodbu modela jezera podataka prema već uspostavljenim formalnim okvirima za rad s prostorno-vremenskim tokovima podataka.

## Doprinosi

Znanstveni doprinos ovog rada sastoji se u sljedećem:

1. Model unificiranog sloja pristupa podacima u jezerima podataka koja uključuju prostorno-vremenske tokove podataka

2. Nadogradnja modela jezera podataka tipovima podataka za potporu prostorno-vremenskim tokovima, kroz unificirani sloj pristupa podacima

3. Algoritmi potpore operacijama nad tipovima prostorno-vremenskih tokova podataka i njihova ugradnja u unificirani sloj pristupa podacima u jezeru podataka

## Struktura rada

Struktura ovog rada je sljedeća:

Poglavlje 1 prikazuje postojeće poznate definicije jezera podataka i predlaže se proširena definicija na temelju analiziranih arhitektura i namjena jezera podataka. Analizirane arhitekture međusobno su uspoređene te su im analizirane prednosti i mane modela. U nastavku poglavlja, sistematizirani su izvori podataka prema strukturi, nakon čega je posebna pažnja posvećena tokovima podataka. Zatim je dan pregled poznatog modela međusloja pristupa podacima na kojem se formalno temelji unificirani sloj pristupa podacima. Konačno, sistematizirani su i analizirani pristupi pohrani podataka u jezeru podataka ovisno o strukturi.

Poglavlje 2 sistematizira postojeća istraživanja u domeni jezera podataka, uvođenja procesa upravljanja podacima u jezerima podataka te različite pristupe izgradnje Lambda i Kappa arhitektura unutar jezera podataka. Nastavno na upravljanje procesima, dan je pregled postojećih poznatih istraživanja i modela metapodataka u jezerima podataka. Poglavlje završava pregledom definicija vezanih za polustrukturirane podatke.

U poglavlju 3 predočeni su svi doprinosi ovog rada. Poglavlje započinje identifikacijom postojećih sustava tipova podataka i prijedlogom proširenja za potporu radu s nestrukturiranim podacima i operatorima nad proširenim skupom tipova podataka. Zatim je predložen dorađeni logički model jezera podataka, uz poseban opis svakog od slojeva. Ključni dio poglavlja je definicija unificiranog sloja pristupa podacima, koje se temelji na prethodno definiranom modelu jezera podataka i skupu tipova podataka i operacija nad njima. Budući da se unificirani sloj pristupa podacima oslanja na repozitorij metapodataka, opisane su njegove klase i uloga, nakon čega je dan popis definiranih algoritama ugrađenih u unificirani sloj pristupa podacima.

Poglavlje 4 sadrži pregled alata za naprednu analizu podataka, podijeljenih u grupe ovisno o zajedničkim značajkama pojedinih alata.

U poglavlju 5 prikazan je prototip modela jezera podataka, s primjerima izgrađenih podatkovnih cjevovoda i povezanih metapodataka. Ujedno je prikazan opis implementacije sloja pohrane podataka i sloja obrade podataka, nakon čega je prototip verificiran kroz primjere podataka od prihvata podataka do njihove vizualizacije.

Rad je zaključen s poglavljem 6 u kojem su dani zaključci istraživanja i smjernice za njegov nastavak.

# 1. Jezera podataka

## 1.1. Uvod

Razvoj velikih podataka doveo je do zahtjeva za analitičkim sustavima koji su horizontalno i vertikalno skalabilni, pružaju potporu za istovremeni rad s heterogenim podacima i izvorima podataka koji generiraju podatke u velikom volumenu (eng. *volume*) i velikom brzinom (eng. *velocity*). Podatke koji dolaze velikom brzinom, u velikom volumenu i velikoj različitosti (eng. *variety*) jednim imenom nazivamo velikim podacima, a spomenute odlike objedinjene su u sintagmi 3V. Postojeća analitička rješenja, primarno gledajući skladišta podataka, do razvoja jezera podataka, mogla su podržati te zahtjeve u određenoj mjeri, no ne i u potpunosti.

Analitički sustavi, do pojave jezera podataka, pretežno su bili temeljeni na relacijskim bazama podataka. Za relacijske baze podataka, što uključuje i skladišta podataka, u slučaju velikih podataka značajnu prepreku predstavlja izvedba u obliku raspodijeljenih sustava. Skladišta podataka, kao podloga za analizu strukturiranih integriranih podataka, nisu prilagođena za rad s velikim podacima, koji su pretežno nestrukturirani [1] i u određenoj mjeri polustrukturirani. Pokušaj prevladavanja takvih prepreka predstavljaju masivni paralelni sustavi za obradu podataka (eng. *massively parallel processing*) koji su omogućili skalabilnost i otpornost na greške [2]. No, potpora za rad s heterogenim podacima i dalje nije ostvarena, budući da se oni moraju korištenjem procesa integracije svesti na zajednički oblik.

U [3] podatkovni cjevovodi definirani su kao spojeni lanac procesa u kojima je jedan ili više procesa izvor podataka za sljedeći proces u nizu. Koriste se za prijenos podataka između izvora podataka i njihovih pohrana, analizu podataka ili spajanje podataka. Uloga podatkovnih cjevovoda, pored navedenih, također je i potpora automatizaciji procesa prikupljanja i pohrane podataka iz različitih izvora.

Jezera podataka prevladavaju izazove velikih podataka postavljene pred skladišta podataka. Prvenstveno, jezera podataka u prihvatu podataka ne zahtijevaju integraciju kako bi se podaci pohranili u jedinstvenom obliku, već se podaci primarno pohranjuju u svojem izvornom, nepromijenjenom obliku. Ujedno, to znači da je proces integracije značajno pojednostavljen, odnosno dominantno je gutanje podataka (eng. *data ingestion*). Proces integracije podataka u skladištima podataka predstavlja najveće procesno opterećenje koje čini stvarnovremensku analizu podataka izuzetno teško izvedivom. Za potporu stvarnovremenskim skladištima podataka nužno je osigurati posebne tehnike, poput integracije mikroserija podataka (eng. *microbatching*).

Podatkovni cjevovodi koji obavljaju obradu, pripremu, transformaciju i obogaćivanje podataka odnose se na integraciju podataka. Takvi podatkovni cjevovodi poznati su i kao ETL (*Extract-Transform-Load*) cjevovodi i služe kao potpora učitavanju serija podataka u analitičke sustave, najčešće u skladišta podataka. S druge strane, kod izvora podataka koji neprekinuto stvaraju nove zapise, odnosno tokove podataka, podatkovni cjevovodi nazivaju se ELT (*Extract—Load- Transform*) cjevovodima i osnova su potpore obradi tokova podataka.

Važno je napomenuti da unutar podatkovnih cjevovoda procesi ne prestaju pohranom podataka. Naprotiv, nakon procesa pohrane podataka, moguće je te podatke nastaviti koristiti u sljedećim definiranim procesima, čime se kompleksne obrade podataka ulančavaju u jedinstvenu sekvencu. Ovakav pristup omogućuje definiranje gutanja podataka i povezanih preslikavanja u nove modele unutar jednog procesa.

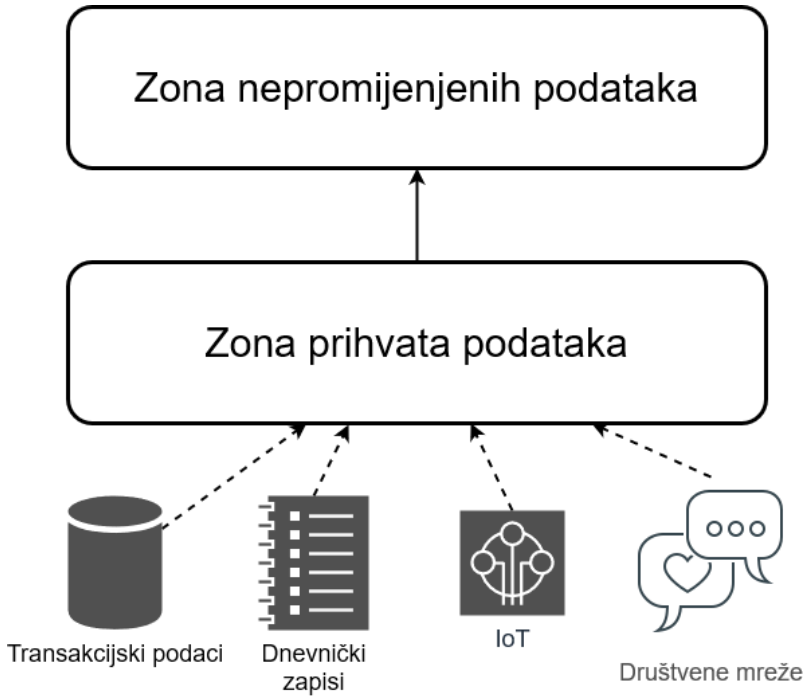
U slučaju jezera podataka, potpora stvarnovremenskoj analizi podataka predstavlja jednu od bitnih odlika koje ono sadrži. Budući da se podaci gutaju, ne postoji dodatni trošak kojeg integracija podataka predstavlja i podaci su u gotovo stvarnom vremenu dostupni za analizu. Uz to, povećanje volumena ili brzine dolaznih podataka ne mora nužno utjecati na degradaciju potpore za stvarnovremensku analizu, već se može rješavati skaliranjem cijelog sustava.

Uzevši sve navedeno u obzir, može se zaključiti kako su jezera podataka u svojoj naravi sustavi za pohranu heterogenih podataka. Podaci pohranjeni u jezeru podataka imaju jednaku strukturu kakva je stigla iz izvora podataka, a jednom pohranjeni podaci više se ne mijenjaju niti brišu.

Na tom tragu, dan je prijedlog osnovne definicije jezera podataka koji se ujedno temelji na radu [41]:

***Definicija 1.*** *Jezero podataka definirano je kao repozitorij heterogenih podataka pohranjenih u izvornom, nepromijenjenom obliku.*

Model jezera podataka koji proizlazi iz Definicija 1 prikazan je na slici Slika 1. Dvozonska arhitektura jezera podataka




Slika 1. Dvozonska arhitektura jezera podataka

Kako je u svojoj osnovi jezero podataka repozitorij nepromijenjenih podataka, svi perzistentno pohranjeni podaci mogu se predočiti kao zona u kojoj su oni dostupni za analize na zahtjev. Time je dana osnova za dvozonski model jezera podataka, u kojem su definirane zone prihvata podataka i zona nepromijenjenih podataka. Zona prihvata podataka predstavlja zonu potpore gutanju podataka u kojoj se podaci nalaze samo tijekom procesa gutanja. Nakon pohrane u perzistentnu memoriju, podaci se miču iz zone prihvata podataka. Perzistentnu memoriju pak predstavlja zona nepromijenjenih podataka, gdje su jednom dohvaćeni i progutani podaci pohranjeni u svom izvornom, nepromijenjenom obliku.

Na temelju dvozonske arhitekture razvili su se kasniji model jezera podataka, analizirani u poglavlju 1.2.

Dvozonska arhitektura jezera podataka bliska je dvoslojnoj arhitekturi skladišta podataka, pri čemu je zona prihvata podataka analogna pripremnom području, a zona nepromijenjenih podataka samom skladištu podataka. Ujedno su i prisutne značajne razlike u pogledu na podatke. Jezero podataka u dvozonskoj arhitekturi nije subjektno orijentirano, odnosno pohranjeni podaci mogu služiti kao potpora analizi u različitim poslovnim područjima. Također, u osnovi je izostavljena komponenta vremenske ovisnosti, budući da podaci u jezeru podataka nemaju nužno pridodanu vremensku dimenziju. Integriranost, kako je ranije spomenuto, u suprotnosti je s osnovnom namjenom jezera podataka kao repozitorija

heterogenih podataka. Kako je ranije navedeno, jednom pohranjeni podaci u jezeru podataka ne mogu se promijeniti, stoga je povijesna nepromjenjivost zajednička osobina.

Generalno gledajući, skladišta podataka i jezera podataka dijele određeni broj značajki. Oba se promatraju kao središnje mjesto za pohranu podataka prikupljenih iz izvornih sustava, osnova su šireg analitičkog sustava i smatraju se jedinstvenim izvorom podataka. Posljednje navedena značajka direktna je posljedica osobine povijesne nepromjenjivosti. Ujedno, u pogledu jedinstvenog izvora podataka, skladišta podataka sadrže obrađene i prilagođene podatke, stoga se ona smatraju jedinstvenim izvorom istinitih podataka. Jezera podataka, s druge strane, ne moraju nužno sadržavati istinite podatke, već samo nepromijenjene izvorne podatke, što ih čini jedinstvenim izvorom činjeničnih (eng. *factual*) podataka. Ovdje treba istaknuti kako pojam činjeničnih podataka nema nikakvu korelaciju s činjeničnim tablicama u dimenzijskom modelu skladišta podataka, već se odnosi na samo značenje pohranjenih podataka.

Skladišta podataka, u pogledu zrelosti sustava, i dalje su ispred jezera podataka. Tome svakako ide u prilog njihova namjena, koja je više orijentirana prema potpori poslovanju i donošenju odlika. Jezera podataka imaju širu ulogu od skladišta podataka i samim time ih očekuje duži evolucijski put do razine zrelosti koju trenutne posjeduju skladišta podataka.

Striktno uzimajući definiciju 1 u obzir, jezero podataka ne bi smjelo sadržavati podatke u obliku prikladnijem za daljnju analitičku upotrebu. Premda primarna svrha korištenja jezera podataka nije potpora tradicionalnim analitičkim i vizualizacijskim alatima, u takvim scenarijima pohrana podataka u obliku prilagođenom za analitiku može donijeti određenu korist. Kao primjer takvih arhitektura mogu se uzeti [42] i [43]. Stoga prethodnu definiciju proširujemo na sljedeći način:

***Definicija 2.*** *Jezero podataka definira se kao repozitorij heterogenih podataka pohranjenih u izvornom, nepromijenjenom obliku te ostalim oblicima prilagođenima za daljnju upotrebu.*

Na ovaj način eksplicitno je navedeno kako se podaci u jezero podataka nužno pohranjuju u izvornom obliku, no ostavljena je i mogućnost višestruke pohrane istih podataka u različitim oblicima. Na taj način, slijedi se paradigma iz [42] i [43], gdje su podaci pohranjeni u prilagođenim oblicima ovisno o zonama. Uzevši u obzir da podaci s izvora podataka dolaze u strukturiranim, polustrukturiranim i nestrukturiranim oblicima, potrebno je omogućiti pohranu podataka u bilo kojem od tri oblika, neovisno o izvornoj strukturi.

Jezera podataka slijede paradigmu obrade ekstrakcija – pohrana – transformacija (ELT), odnosno njezin prošireni oblik ekstrakcija – pohrana – transformacija – analiza (eng. *extract – load – transform – analysis*, skraćeno ELTA). Za razliku od skladišta podataka, koja u svrhu poboljšanja performansa dohvata podataka prethodno odrađuju proces transformacije i takve podatke pohranjuju u postrelacijsku bazu podataka, jezera podataka oslanjaju se na raspodijeljene sustave koji omogućuju brzi dohvat i transformaciju podataka kroz paralelne zadatke obrade.

Iako su pohrana, dohvat i obrada podataka znatno brži u raspodijeljenim sustavima, oni ne osiguravaju stalnu konzistentnost podataka. Naprotiv, u slučaju raspodijeljenih sustava radi se o konačnoj konzistentnosti – što kod analitičkih upita može dovesti do situacija kada isti upit neće rezultirati istim skupom podataka. U tu svrhu, u prijedlog modela jezera podataka uvodi se pohrana potrebnog skupa strukturiranih zapisa u postrelacijsku bazu podataka, kako bi se osiguranje konzistentnosti podataka provelo kroz sustav za upravljanje relacijskim bazama podataka. Nužno je istaknuti kako pohrana skupova podataka u postrelacijske baze podataka ne izostavlja pohranu nepromijenjenih zapisa u sam repozitorij.

Pored postrelacijskih baza podataka, u prijedlog modela jezera podataka uvodi se i pohrana u NoSql baze podataka. Kombinacija više vrsta NoSql baza podataka može se iskoristiti za implementaciju algoritama temeljenih na odnosu podataka na način da se podaci ne dohvaćaju u izvornom obliku, već se koriste prilagođeni, polustrukturirani i strukturirani zapisi. Kako se radi o transformiranim podacima, izvedba algoritma ne zahtijeva korak transformacija za svaki dohvaćeni podatak, što ubrzava njegovu izvedbu.

Inicijalno, arhitekture jezera podataka temeljile su se na pohrani nepromijenjenih podataka direktno u datotečne sustave. Podaci su potom dohvaćeni kroz odgovarajuće alate, u kojima je njihova struktura bila definirana ovisno o konačnoj namjeni i potom konzumirana. Pritom podaci u nijednom trenutku nisu imali definirane čvrste strukture. Razvojem i uvođenjem novih modela jezera podataka, takav pristup se promijenio. Veliki pomak u paradigmi struktura podataka unutar jezera podataka napravljen je uvođenjem zonske arhitekture [42], koja sadrži zone u kojima podaci uvijek imaju poznatu strukturu. Poznate strukture omogućavaju pohranu podataka korištenjem shema i relacija u dobro poznate sustave za pohranu podataka. Takvi sustavi, pored pohrane, podržavaju velik broj optimiziranih operacija nad podacima i ugrađenih analitičkih funkcija, što može olakšati rad s podacima u pojedinim zonama jezera podataka. Značajnu ulogu pritom imaju i metapodaci, pomoću kojih se takve poznate strukture opisuju.

Prolaskom kroz zone u jezeru podataka, strukture zapisa mogu se mijenjati, a zapisom slijeda promjena podržava se praćenje porijekla podataka od njihovih izvora do pohrane i korištenja. Modeli baza podatka korišteni u predloženom modelu jezera podataka detaljnije su opisani kasnije u poglavlju 1.2.

Unutar jezera podataka metapodaci generalno imaju važnu ulogu. Iako su metapodaci poznati u postrelacijskim bazama podataka, kao i u skladištima podataka, njihova uloga posebno je istaknuta u jezerima podataka. Budući da su u jezerima podataka pohranjene velike količine heterogenih podataka, za njihovu kasniju analizu i obradu nužno je kvalitetno ih opisati metapodacima. Idealni slijed obrade podataka u jezerima podataka uključuje pohranu metapodataka o novim izvorima podataka do razine pojedinog atributa, no ključno je da oni budu pohranjeni prije prvog dohvata podataka iz izvora. Također, korištenjem procesa ekstrakcije metapodataka u naprednijim modelima jezera podataka moguće je automatizirati detekciju i pohranu metapodataka iz samih podataka.

Održavanje metapodataka jedan je od temelja uspješne implementacije procesa upravljanja podacima unutar jezera podataka. Kako su jezera podataka bila smatrana repozitorijem podataka, u samim začecima nedovoljno se isticao značaj cjelovitog upravljanja podacima. Kvalitetni repozitoriji metapodataka pritom imaju veliku ulogu u uspješnosti upravljanja podacima, budući da osim interobjektnih, intraobjektnih i procesnih metapodataka mogu biti prošireni kako bi podržali sve tražene cjeline. Primjerice, u repozitorijima metapodataka mogu se definirati razine pristupa podacima na razini pojedinog atributa kroz dodavanje metapodataka o korisničkim računima, grupama i ulogama.

## 1.2. Modeli jezera podataka

U radu [4] identificirana su dva glavna pravca u arhitekturi jezera podataka: arhitekturu jezeraca (eng. *pond*) i zonska arhitektura. Pored njih, u radu [41] predložen je model temeljen na slojevima. Arhitekturu jezeraca zagovara Inmon [40], pri čemu su sastavne jedinice podijeljene u pet cjelina, odnosno jezeraca:

1) Jezerce sirovih podataka (eng. *raw data pond*) sadrži progutane podatke iz izvora u nepromijenjenom obliku. Njegova je namjena u jezeru podataka prihvat podataka iz vanjskih izvora kako bi postali dostupni ostalim jezerima i ne sadrži nikakve metapodatke. U jezercu sirovih podataka oni se zadržavaju do prijenosa u jedno od preostalih jezera.
2) U jezerce analognih podataka (eng. *analog data pond*) pohranjuju se podaci, uglavnom polustrukturirani, koji pristižu velikom brzinom iz izvora podataka. Izvor podataka za jezerce analognih podataka (ali i za aplikacijsko i tekstualno jezerce podataka) je jezerce sirovih podataka.
3) Aplikacijsko jezerce podataka (eng. *application data pond*) u suštini predstavlja skladište podataka u kojem se nalaze strukturirani podaci, prethodno provedeni kroz procese integracije (ekstrakciju, transformaciju i učitavanje) podataka.
4) Tekstualno jezerce podataka (eng. *textual data pond*) sadrži nestrukturirane tekstualne zapise, pogodne za daljnju analizu teksta i postizanje kontekstualizacije. U tu svrhu koristi se tekstualna višeznačnost (eng. *textual disambiguation*), odnosno tekstualni ETL proces. Tekstualna višeznačnost označava proces prilikom kojeg se postojeći tekstualni zapisi proširuju dodatnim tekstualnom podacima, proširujući izvornu informaciju.
5) Arhivsko jezerce podataka (eng. *archival data pond*) sadrži podatke koji se trenutno ne koriste aktivno u ostalim jezercima. Podaci u njega pristižu iz tri jezerca u kojima su podaci prethodno poprimili neku strukturu.

U Inmonovom modelu, podaci se fizički odvajaju u jezera prema njihovoj strukturi i namjeni. Kao konačna struktura predviđen je relacijski (strukturirani) oblik, temeljeno na zaključku kako većina analitičkih i vizualizacijskih alata podržava relacijski oblik podataka. Na taj način omogućena je brza analiza podataka. Također, pri analizi podataka iz više jezera, nužno je korištenje metapodataka kako bi proces analize mogao biti obavljen nad svim podacima. Najveći mu je svakako nedostatak perzistentne pohrane podataka u izvornom obliku, što potencijalno u konačnici može dovesti do izostanka informacija prilikom analize podataka.

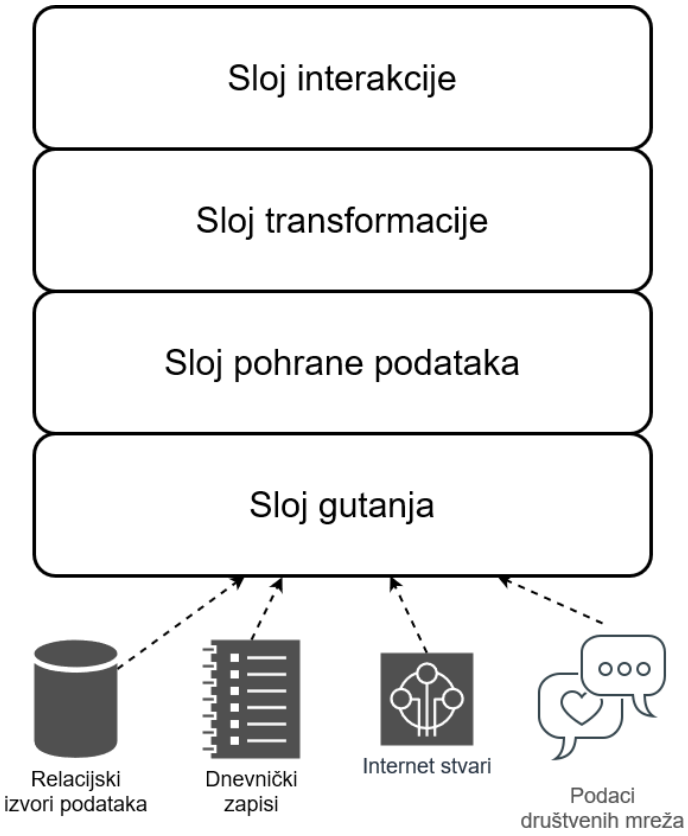
Drugi pristup arhitekturi ponuđen je u [41] i temelji se na slojevima različitih zadataka i prikazan je na slici Slika 2. Slojevita arhitektura jezera podataka [41]. Predložena su četiri sloja:

1) Sloj gutanja (eng. *ingestion layer*) sa zadaćom gutanja podataka iz heterogenih izvora podataka u jezero podataka. U njemu se, pored gutanja sirovih podataka, ujedno odrađuje i ekstrakcija metapodataka pomoću automatskih procesa nad strukturiranim i polustrukturiranim podacima i njihova pohrana u repozitorij metapodataka.

2) Sloj pohrane podataka (eng. *storage layer*) sadrži repozitorij metapodataka i repozitorije sirovih podataka. Repozitoriji sirovih podataka moraju pružati potporu svim oblicima i strukturama zapisa podataka, što uključuje i pohranu datoteka te pohranu zapisa u nepromijenjenom obliku. Kompleksnost pohrane potrebno je od krajnjih korisnika sakriti sučeljem koje pruža mogućnost izvođenja upita.

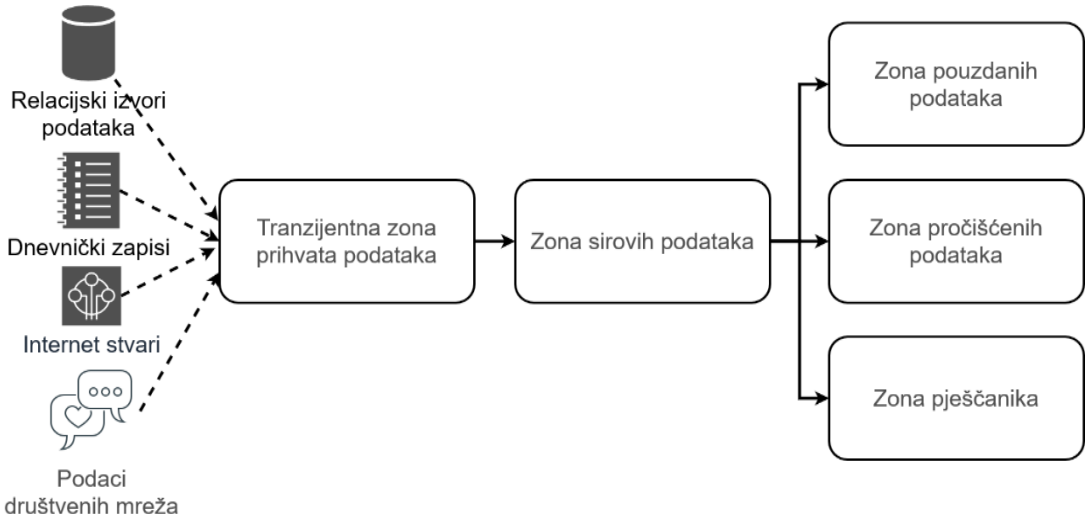
3) Sloj transformacije (eng. *transformation layer*) pruža mogućnost skalabilnog izvršavanja operacija čišćenja, transformacija i integracije podataka kako bi poprimili odgovarajući konačni oblik. U sloju transformacije stvaraju se modeli slični područnim skladištima podataka (eng. *data mart*). Sve novodetektirane informacije prilikom izvođenja transformacija upisuju se u repozitorij metapodataka.

4) Sloj interakcije (eng. *interaction layer*) daje korisnicima pristup repozitoriju metapodataka i transformiranim podacima iz sloja transformacije u svrhu istraživanja podataka, izvođenja analitičkih upita i vizualizacija pomoću odgovarajućih alata.




Slika 2. Slojevita arhitektura jezera podataka [41]

Konačno, jedan od često korištenih modela jezera podataka u posljednje vrijeme, temeljen na zonskoj arhitekturi, predstavljen je od strane kompanije Zaloni u radu [42]. Predloženi model koristi četiri osnovne zone i zonu pješčanika, u kojima se podaci prate, validiraju, pročišćuju te im se dodjeljuju metapodaci. Arhitektura tog modela prikazana je na slici 3.




Slika 3. Zaloni model jezera podataka [42]

Prva zona naziva se tranzijentna zona prihvata podataka (eng. *Transient Landing Zone*) i sadrži privremeno pohranjene podatke, dostavljene u izvornom obliku. Tranzijentna zona prihvata podataka je pristup korišten u skladištenju podataka i obuhvaća prostor dostave datoteka i podataka iz kojeg se podaci konzumiraju i po završetku procesa obrade ili gutanja brišu. U ovoj zoni radi se preliminarna analiza podataka te detekcija potencijalnih tehničkih ili poslovnih nesukladnosti podataka, što je ujedno jedan od preduvjeta uspostave uspješnog procesa upravljanja podacima. Tranzijentna zona prihvata podataka poželjna je u modelu jezera podataka, no nije nužna ako njen izostanak ne utječe na zadane procese upravljanja podacima.

U zoni sirovih podataka (eng. *Raw Zone*) nalaze se nepromijenjeni podaci dohvaćeni iz tranzijentne zone slijetanja. U zoni sirovih podataka trajno se pohranjuju podaci u nepromijenjenom, izvornom obliku. Time zona sirovih podataka predstavlja jedinstveni izvor pouzdanih podataka za analizu i daljnju obradu. U ovoj zoni radi se prva obrada podataka, što rezultira indeksiranjem podataka i proširivanjem zapisa odgovarajućim metapodacima. U samoj zoni ne očekuju se provjere kvalitete i usklađenosti podataka, već se podrazumijeva da su nužni dijelovi procesa prethodno provedeni. Podaci iz zone sirovih podataka mogu se koristiti kao izvor u cjevovodima podataka, kao i za analitičke potrebe direktnim dohvatom iz same zone.

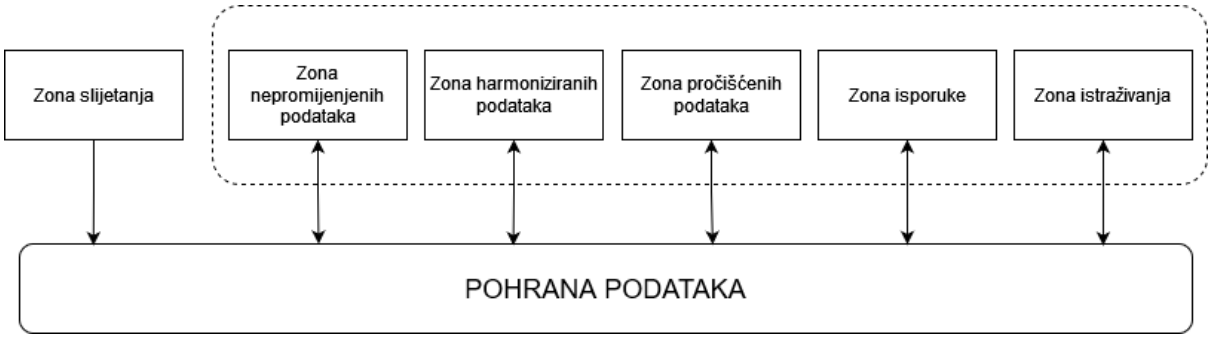
Zona pouzdanih podataka (eng. *Trusted Zone*) sadrži podatke nad kojima se provode dodatne provjere usklađenosti i kvalitete podataka, ovisno o konačnoj namjeni. Izvor podataka za zonu pouzdanih podataka je zona sirovih podataka, iz koje se dohvaćaju tehnički i regulatorno ispravni podaci. Unutar zone pouzdanih podataka, primjenom procesa čišćenja i validacije podataka, stvaraju se novi skupovi obrađenih podataka pretvorenih u oblike pogodne za poslovno korištenje. Zona pouzdanih podataka može sadržavati glavne podatke (*eng. master data*) i referencijske podatke (*eng. reference data*) koji su dio procesa upravljanja podacima. U ovoj zoni, glavni podaci predstavljaju osnovne skupove podataka s namjenom opisa podataka, a koji su povezani putem određenih domenskih vrijednosti ili hijerarhija. Referencijski podaci, s druge strane, obuhvaćaju više skupova glavnih podataka spojenih u jedinstveni zapis koji se odnosi na pojedini opisani entitet među podacima. Uloga referencijskih podataka može se predočiti kao pojedinačna verzija istinitih podataka za svaki opisani entitet unutar zone.

Zona pročišćenih podataka (eng. *Refined Zone*) izvor je podataka za korisnike koji smiju imati pristup samo određenom podskupu podataka. Opseg pristupa određuje se kroz odgovarajuće procese upravljanja podacima, dodjeljivanjem skupa prava pristupa za pojedinog

korisnika ili grupe korisnika. U zoni pročišćenih podataka nalaze se podaci oblikom prilagođeni krajnjim korisnicima, nad kojima su izvršene promjene kako bi odgovarale specifičnim korisničkim poslovnih zahtjevima. Također, namjena zone pročišćenih podataka jest pružiti potporu za izgradnju modela podataka nad kojima se izvršavaju upiti od strane korisničkih aplikacija ili samih korisnika.

Konačno, zona pješčanika (eng. *Sandbox*) služi za istraživanja i ad-hoc analize nad podacima. Navedena zona pretežno ima funkciju testnog okruženja u kojem se podaci dohvaćaju iz svih preostalih zona, no može služiti kao novi izvor podataka za samo jezero podataka ili kao dodatni izvor podataka u podatkovnom cjevovodu.

U [43] dan je prijedlog modela jezera podataka izveden iz [42], temeljen na modelu trezora podataka (*eng. data vault*). Navedeni model koristi, pored zone slijetanja podataka, pet organizacijskih zona podataka, u kojima su oni prilagođeni svojoj konačnoj namjeni, kako je prikazano na slici 4:




Slika 4. Organizacijske zone modela jezera podataka iz [43]

U zoni nepromijenjenih podataka pohranjuju se podaci u nepromijenjenom obliku. Time se osigurava uvjet povijesne nepromjenjivosti podataka, te ujedno jezero podataka služi kao jedinstveni izvor za analize i obrade podataka.

Zona harmoniziranih podataka i zona pročišćenih podataka modelirane su po načelima trezora podataka. U njima se nalaze strukturirani podaci, pri čemu zona harmoniziranih podataka sadrži sirove podatke u strukturiranom obliku, dok zona pročišćenih podataka nad njima primjenjuje prethodno definiranu poslovnu logiku. Zona harmoniziranih podataka svojom namjenom odgovara zoni pouzdanih podataka u modelu [42], dok je zona pročišćenih podataka identična namjeni istoimenoj zoni u [42].

Zona isporuke i zona istraživanja nemaju definiranu strukturu, već se ona definira prema potrebama pojedinih korisnika i alata. Obje zone izvedene su iz zone pješčanika predložene u

[42], no u ovom modelu podaci su odvojeni prema krajnjoj namjeni. U zoni isporuke tako se nalaze podaci koji su prethodno obrađeni za korištenje kroz specifične alate, dok model za podataka u zoni istraživanja definiraju sami korisnici kroz ad-hoc analize.

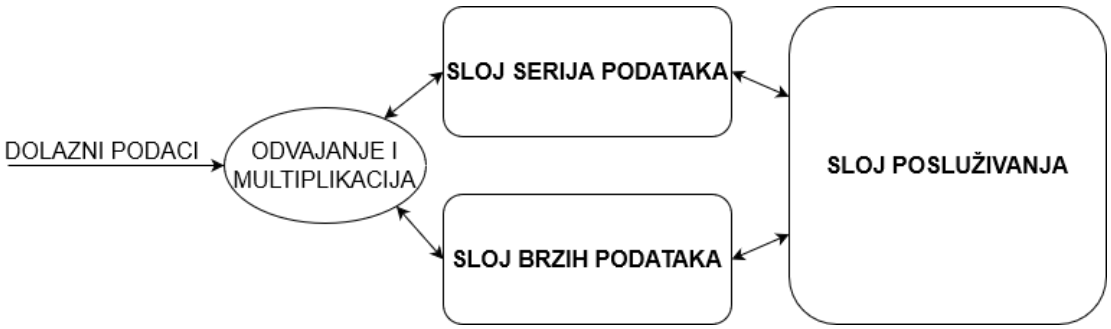
Sve navedene zone predstavljaju organizacijski sloj iznad pohrane podataka i predstavljaju sučelja jezera podataka prema korisnicima i sustavima van samog jezera podataka.

Organizacija podataka u zonama direktno je oslonjena na repozitorij metapodataka u kojem se definiraju strukture i međuovisnosti objekata i atributa. Po [43], barem jedna zona mora sadržavati strukturirane podatke, što je diktirano korištenjem trezora podataka. Zone podataka su međusobno neovisne i zasebno definiraju tehniku modeliranja podataka. Ujedno, korištenje pojedine zone nameće i način pohrane podataka, stoga model direktno utječe na izbor pojedinih oblika pohrane. Iz modela vidljivo je da je pohrana nestrukturiranih podataka nužna za potporu predloženoj organizaciji podataka, dočim je pohrana strukturiranih i polustrukturiranih podataka ovisna o odabiru i modelu u pojedinoj zoni podataka.

Navedeni model ima višestruku primjenu, od obrade i pripreme podataka tijekom dohvata, preko potpore poslovnim korisnicima pomoću već obrađenih podataka do pružanja potpore naprednoj analitici kroz prilagođene podatke. Zona nepromijenjenih podataka može biti korištena kao međukorak u obradi podataka ili dio podatkovnog cjevovoda. Korištenjem te zone ostvaruje se potpora obradi tokova podataka nakon što su njihovi podaci pohranjeni u trajnu memoriju. Zona harmoniziranih i zona pročišćenih podataka u predloženoj organizaciji prvenstveno su potpora poslovnim korisnicima i sustavima koji se oslanjaju na strukturirane podatke. Zona isporuke i zona istraživanja namijenjene su prilagodbi podataka za daljnje obrade i analize. Pritom valja istaknuti da u njima ne postoji striktno definirana struktura, već se ona definira prema određenim potrebama. Stoga se u njoj u predloženom modelu mogu pronaći strukturirani, nestrukturirani i polustrukturirani podaci koji mogu prethodno biti obrađeni i pohranjeni u tom obliku.

Lambda arhitektura uvedena je u jezera podataka kao premosnica između dva pristupa učitavanju i obradi podataka: serijskom i gotovo stvarnovremenskom. Iako u kontekstu serijskog učitavanja radimo distinkciju u odnosu na serijsko učitavanje u skladišta podataka, gdje je latencija između pojave podataka na izvoru i njihove pohrane u pravilu više sati do jednog dana, određeni vremenski odmak između prihvata i učitavanja podataka i dalje je prisutan.

Lambda arhitektura sastoji se od tri sloja: sloja serija podataka (eng. *batch layer*), sloja brzih podataka (eng. *speed layer*) i sloja posluživanja (eng. *serving layer*) [18] [19]. Shema Lambda arhitekture prikazana je na slici 5



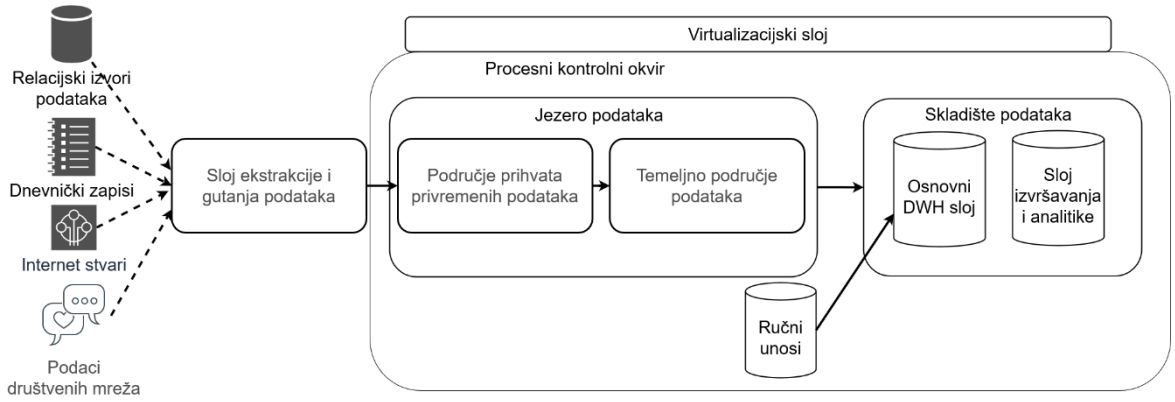

Slika 5. Lambda arhitektura

Serijska obrada i učitavanje podataka unutar jezera podataka temelji se na obradi manjih serija podataka, gdje volumen podataka određuje veličine same serije. Podaci se, slično kao i u skladište podataka, pohranjuju u nepromjenjivi skup podataka u kojem se podaci isključivo nadodaju [18]. Na temelju navedenog skupa podataka, kreiraju se prilagođeni pogledi koji se periodički osvježavaju nakon obrade i pohrane nove serije podataka. Negativna strana, kao što je navedeno u [19], jest da obrada i pohrana jedne serije podataka, ovisno o kompleksnosti, može trajati satima.

Zone slijetanja i zona nepromijenjenih podataka osnova su predloženih modela Lambda arhitekture i predstavljaju osnovnu podjelu podataka bez dodatne segmentacije prema načinu korištenja.

Kao posebnu arhitekturu jezera podataka važno je istaknuti jezersko skladišta podataka (eng. *Data Lakehouse*) [61], nastalo iz potrebe za integracijom poznatog analitičkog rješenja (skladišta podataka) i mogućnosti pohrane i analize velikih skupova heterogenih podataka (jezero podataka). Jezersko skladište podataka zamišljeno je kao nadogradnja jezera podataka, koje istovremeno služi kao izvor nepromijenjenih podataka za alate napredne analize i izvor podataka za procese integracije podataka u skladište podataka.

Jezersko skladište podataka, pored jezera podataka i skladišta podataka, uključuje i virtualizacijski sloj te sustav upravljanja procesima. Arhitektura jezerskog skladišta podataka prikazana je na slici 6:




Slika 6. Arhitektura jezerskog skladišta podataka (preuzeto iz [61])

Jezersko skladište podataka nema eksplicitno definirane slojeve niti zone podataka. Model jezera podataka, predložen unutar arhitekture jezerskog skladišta podataka, odnosi se na osnovni dvozonski model. Povezivanjem jezera podataka i skladišta podataka kreirana je arhitektura bliska modelu predloženo u [42], iako bez implicitno definiranih zona. Pritom jezero podataka sadrži tranzijentnu zonu slijetanja podataka i zonu nepromijenjenih podataka, a skladište podataka predstavlja zone pouzdanih i zone pročišćenih podataka. Sloj virtualizacije, koji ima pristup podacima i u jezeru i u skladištu podataka, može se poistovjetiti sa zonom pješčanika.

## 1.3. Usporedba analiziranih modela jezera podataka

Inmonov model jezera podataka fokusiran je na strukturu podataka i procesno je nastavak filozofije integracije podataka i njihovog strukturiranja. Kako su podaci logički razdijeljeni u jezerca, ovisno o strukturi, njihovo objedinjavanje prilikom korisničkih upita zahtjeva dodatni međusloj. U njemu se podaci spajaju u jedinstveni rezultat na temelju metapodataka, no sam međusloj nije naveden u modelu jezera podataka.

Najveći nedostatak Inmonovog modela leži u činjenici da ne postoje sirovi povijesni podaci. Sirovi podaci nakon prijenosa u odgovarajuće jezerce brišu se iz jezerca sirovih podataka i povijesni zapis podataka ovisi o izvršenoj transformaciji prilikom prijenosa u jezerca. U slučaju da se transformacija promijeni, nije moguće promjenu primijeniti na povijesne podatke. Time dolazi do rizika nemogućnosti korištenja povijesnih podataka, a samim time i gubitka vrijednih informacija. Jedan od pristupa rješavanju ovog rizika jest dodavanje dodatnog jezerca s ulogom perzistentne pohrane sirovih podataka. Time se mijenja podatkovni cjevovod između jezeraca – istovremeno bi se punila sva jezerca iz jezerca sirovih podataka, a u slučaju promjene transformacije moguće je ponovno obraditi povijesne sirove podatke i na taj način ažurirati postojeće podatke u jezercima. Također, tada se potencijalno iz arhitekture može izostaviti arhivsko jezerce podataka – sve nekorištene podatke moguće je ponovno obraditi i prenijeti ih u ostala jezerca.

Zonska arhitektura i arhitektura temeljena na slojevima komplementarne su po pitanju izvedbe modela. U zonskoj arhitekturi fokus je na strukturi podataka i podatkovnom cjevovodu između zona, od prihvata sirovih podataka do njihove prezentacije krajnjim korisnicima. Istovremeno, način pohrane podataka nije eksplicitno definiran. Arhitektura temeljena na slojevima veći fokus stavlja na uloge pojedinog dijela jezera podataka, neovisno o strukturi podataka i krajnjim korisnicima.

Tranzijentna zona prihvata podataka i sloj gutanja podataka dijele zadaću prihvata i pripreme za učitavanje podataka u jezero podataka. Prvenstveno, sličnost im je u činjenici da su obje prolazne zone u kojima se podaci iz izvornih sustava ne zadržavaju. Također, dijele zadaću osnovne kontrole kvalitete podataka, pri čemu se podaci segmentiraju u prihvatljive i neprihvatljive podatke za daljnju obradu i pohranu. Na taj način postavlja se osnova za izbjegavanje nepotrebnog gomilanja beskorisnih podataka čak i prije pohrane u jezero podataka. Sloj gutanja podataka ima jednu značajnu razliku – predviđa mogućnost ekstrakcije metapodataka tijekom obrade i njihovu pripremu za pohranu u repozitorij metapodataka.

U zonskoj arhitekturi, zona sirovih podataka sadrži podatke u njihovom izvornom obliku, no bez specifikacije na koji način će se oni pohraniti. S druge strane, sloj pohrane podataka eksplicitno specificira koji su različiti oblici pohrane potrebni za pohranu sirovih podataka, pri čemu ne smije biti izostavljena pohrana nijednog oblika podataka za kojeg se očekuje buduće korištenje. Također, sloj pohrane podataka eksplicitno navodi repozitorij metapodataka kao svoju nužnu komponentu, što nijedna od zona u zonskoj arhitekturi ne specificira kao vlastitu komponentu. Iz ove komparacije može se zaključiti kako je repozitorij metapodataka definiran izvan svih zona, no ujedno im mora biti dostupan.

Zona harmoniziranih i zona pročišćenih podataka pretežno su orijentirane prema krajnjim korisnicima koji koriste podatke u obliku prilagođenom njihovim potrebama. Sličnu ulogu ima i transformacijski sloj, s razlikom što se kod njega podaci prilagođavaju prilikom svakog korisničkog dohvata podataka, dok su u zonskoj arhitekturi strukture podataka prethodno eksplicitno definirane. Zone harmoniziranih i pročišćenih podataka nude prednost bržeg dohvata podataka, budući da su podaci prethodno obrađeni i pohranjeni u definiranom obliku. Uz to, zona pročišćenih podataka posjeduje i komponentu sigurnosti pristupa putem definiranih korisničkih uloga kojima se ograničava dostupnost skupa podataka. Transformacijski sloj ograničava skupove podataka prema definiranim korisničkim zahtjevima, no pritom ne uzima u obzir korisničke uloge, već tu zadaću prepušta aplikacijama izvan jezera podataka.

Transformacijski sloj, zona harmoniziranih podataka i zona pročišćenih podataka imaju sličnosti s područnim skladištima podataka. Kod svih je zajednička uloga stvaranja korisnički orijentiranog ograničenog skupa podataka iz povijesnih podataka, no s bitnom razlikom da su u zonskoj arhitekturi svi podaci prethodno u prilagođenom obliku pohranjeni u jezero podataka. Dodatno, transformacijski sloj dijeli konačnu namjenu sa zonom isporuke iz razloga što su u njima podaci pripremljeni za korištenje u povezanim korisničkim aplikacijama.

Sloj interakcije sličan je zoni istraživanja. Iako se oslanja na transformacijski sloj za pretvorbu korisničkih upita prema pohranjenim podacima u njihovim strukturama, korisnicima je dostupan repozitorij metapodataka u svrhu istraživačkog pristupa podacima. Podaci u sloju interakcije i u zoni istraživanja nemaju konačnu strukturu, već se ona definira prilikom svakog upita i analize. Također, u njima su izloženi sirovi podaci zajedno sa svojim metapodacima kako bi se omogućilo dohvaćanje novih informacija i kreiranje znanja iz podataka kroz stvaranje novih modela podataka i upita nad njima.

## 1.4. Diskusija - Odabir prikladnih postojećih modela jezera podataka

Za odabrane modele jezera podataka iz poglavlja 1.2 u nastavku bit će naveden niz prilagodbi pomoću kojih će u njemu biti dana potpora za prostorno-vremenske tokove podataka.

Kao pogodne arhitekture za prilagodbu modela jezera u svrhu potpore prostorno-vremenskim tokovima podataka odabrane su zonska arhitektura jezera podataka [42] [43] i slojevita arhitektura jezera podataka [41]. Inmonova arhitektura isključena je iz razmatranja, prvenstveno radi manjka kojeg predstavlja izostanak pohrane povijesnih sirovih podataka. Također, iz razmatranja isključena je bazična arhitektura koja uključuje samo zone prihvata podataka i pohrane nepromijenjenih podataka, budući da je ona već proširena kroz arhitekture predložene u [42] i [43].

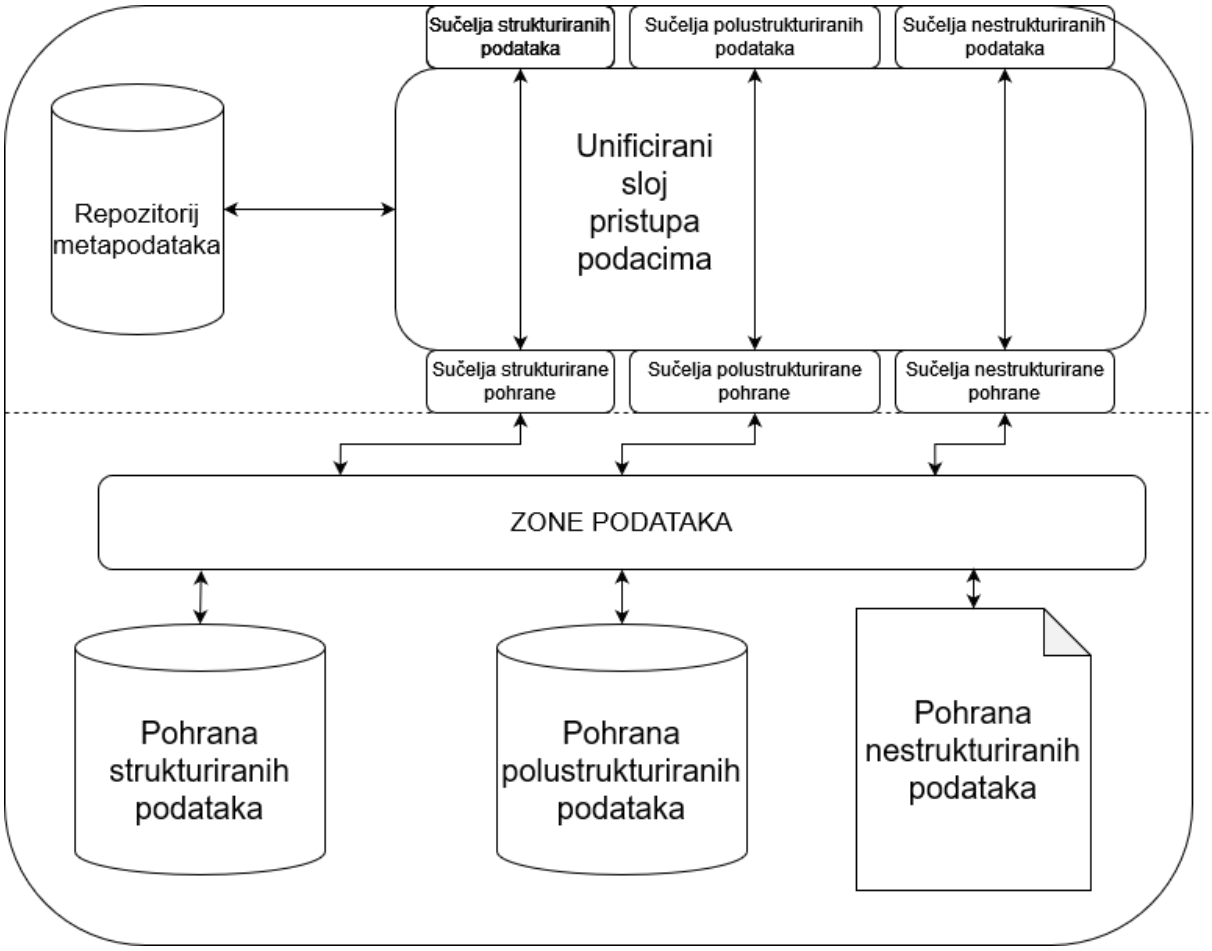
Jedna od ključnih promjena koju je potrebno provesti nad predloženim modelima odnosi se na strukturu zapisa u harmoniziranoj i pročišćenoj zoni. Trezor podataka je prvenstveno tehnika modeliranja primjenjiva u skladištenju podataka, stoga je navedene zone potrebno prilagoditi zahtjevima potpore neovisno o konačnoj namjeni. Korištenje trezora podataka implicira skladištenje podataka i poslovnu inteligenciju, dok namjena jezera podataka ne bi smjela biti ograničavajuća. Potrebno je zadržati njihovu namjenu potpore obrađenim podacima, odnosno njihovom unaprijed zadanom pristupu, no ujedno je nužna i promjena strukture zapisa iz strukturiranih u polustrukturirane. Primarni razlog navedene dorade leži u činjenici da se nestrukturirani i strukturirani zapisi značajno lakše pretvaraju u polustrukturirane no što se polustrukturirani zapisi pretvaraju u strukturirane. Također, veliki broj analitičkih aplikacija, kao i sustava za upravljanje objektno-relacijskim bazama podataka podržava rad s polustrukturiranim zapisima kroz ugrađene funkcije i prilagođene tipove podataka. Iz navedenih razloga, polustrukturirani zapisi ne predstavljaju ograničavajući faktor u prikazu podataka prema vanjskim sustavima.

Spajanjem obje predložene arhitekture omogućuje se detaljno specificiranje dostupnih podataka i njihovih struktura temeljem zonske arhitekture, dok slojevita arhitektura specificira tok podataka već pohranjenih podataka prema korisnicima. Njihovo spajanje rezultira arhitekturom koja definira

- procese gutanja i integracije podataka te njihova pridijeljena sučelja
- načine pohrane podataka
- podatkovni cjevovod između dolaznih i pohranjenih podataka
- zone dostupnosti podataka i njihova pridijeljena sučelja

- smještaj repozitorija metapodataka u modelu jezera metapodataka i njegovu ulogu

kako je prikazano na slici **7**




Slika 7. Spajanje zonske i slojevite arhitekture

Kako su i sami autori naveli u [41], slojevi transformacije i interakcije međusobno su ovisni u podatkovnom cjevovodu. U svrhu objedinjavanja njihove uloge, nameće se rješenje spajanja oba sloja u jedan koji će na temelju korisničkih zahtjeva i upita moći pristupiti prethodno pohranjenim i obrađenim podacima. Svi korisnički upiti i zahtjevi temeljeni su na metapodacima, pohranjenima u repozitoriju metapodataka, koji su korisnicima potrebni u svrhu kreiranja vlastitih podatkovnih modela nad samim podacima.

Sloj gutanja podataka i repozitorij metapodataka u slojevitoj arhitekturi imaju odnos izvora i odredišta podataka. U slojevima transformacije i interakcije podaci se pretežno dohvaćaju iz repozitorija metapodataka, dok sloj gutanja podataka kroz ekstrakciju metapodataka puni repozitorij. Postavke izvornih sustava u sloju gutanja podataka navedene su kao odvojeni zapis, iako su po svojim svojstvima bliske metapodacima te se mogu smatrati dijelom repozitorija

metapodataka kao metapodaci izvora podataka. Analogno, definicije upravljanja kvalitetom podataka oslanjaju se na metapodatke, stoga i njih možemo smatrati posebnim dijelom repozitorija metapodataka.

Sva tri navedena sloja, kao što je vidljivo, oslanjaju se na repozitorij metapodataka kako bi ostvarili tražene funkcionalnosti. Integriranjem slojeva gutanja podataka, transformacije i interakcije stvara se jedinstveni sloj izložen vanjskim sustavima (izvorima podataka, aplikacijama i korisnicima), temeljen na metapodacima. Funkcionalno, takav jedinstveni sloj zadovoljava definiciju integriranih slojeva, budući da se integracijom ne gube njihove inicijalne funkcionalnosti i uloge. Sloj pohrane podataka, pritom, ostaje logički odvojen od ostalih integriranih slojeva.

Integracijom u jedinstveni sloj pruža se mogućnost specificiranja načina pristupa podacima od strane korisnika i aplikacija te specificiranje načina pristupa pohranjenim podacima nakon prilagodbe upita. Kroz specificiranje sučelja kao načina pristupa, jedinstveni sloj pristupa podacima ujedno je približen zonskoj arhitekturi u kojoj pojedina zona ima dodijeljenu ulogu i dostupan skup podataka.

Arhitektura predloženog modela jezera podataka djelomično dijeli paradigmu s Lambda arhitekturom kroz potporu za obradu serija podataka i tokova podataka. Lambda arhitektura [18] [19] [22] pristup je modeliranju obrade i pohrane podatka koja se temelji na podjeli sustava na tri sloja: sloju serija podataka (eng. *batch layer*), sloju brzih podatka (eng. *speed layer*) i uslužnom sloju (eng. *serving layer*). Njihova je namjena razdijeliti podatke na brzo dostupne, manje skupove podataka koji sadrže nedavne podatke i točne, velike skupove podataka u kojima se nalaze povijesni podaci.

Sloj serija podataka sadrži integrirane povijesne podatke pohranjene u obliku prilagođenom sustavima pohrane. U njemu se nalaze multiplicirani podaci u strukturiranim i polustrukturiranim oblicima zapisa, prilagođeni analitičkim upitima i algoritmima za rad s velikim skupovima podataka. U jezera podataka, sljedeći lambda arhitekturu, sloj serija podataka odnosi se isključivo na pohranu podataka. Kako bi se podaci prilagodili upitima i spajanju sa slojem brzih podataka, nakon pohrane podataka sloj serija podataka obavlja dodatne operacije nad podacima.

Uslužni sloj usko se naslanja na sloj serija podataka. Njegova je glavna namjena prikaz podataka u prilagođenom obliku za dohvat kroz upite iz analitičkih alata, bilo da se radi o

klasičnim upitima ili naprednoj analitici. Uslužni sloj tipično sadrži podatke pohranjene u sloju serija podataka u strukturiranom obliku. Podaci se prethodno trebaju prilagoditi strukturiranom obliku, za što se u sloju serija podataka koriste pogledi i posebno modelirani entiteti.

Sloj brzih podataka najviše služi kao potpora stvarnovremenskim procesima. U svrhu smanjenja latencije između prihvata i pohrane podataka, u sloju brzih podataka nalaze se isključivo podaci u nepromijenjenom obliku. Vrijeme dostupnosti podataka u sloju brzih podataka ovisi o procesima obrade i pohrane u sloj serija podataka – po završetku uspješne pohrane podataka ostvaruje se mogućnost uklanjanja podataka iz sloja brzih podataka.

## 1.5. Sistematizacija izvora podataka

Svi izvori podataka mogu se podijeliti prema korištenom obliku strukture u strukturirane, polustrukturirane i nestrukturirane.

### 1.5.1. Strukturirani izvori podataka

Strukturirani podaci mogu sadržavati sve tipove podataka koji postoje u skupu podržanih tipova podataka, navedenima u [32]. Bitna značajka strukturiranih podataka jest da je sama struktura podataka poznata prije no što se sam zapis (ntorka) pojavi u sustavu.

Uz strukturirane podatke, odnosno izvore podataka koji dostavljaju strukturirane podatke, vežu se metapodaci o:

- Izvoru podataka (naziv, opis, ...)
- Entitetima u izvoru podataka (naziv, opis, pripadnost izvoru, ...)
- Strukturi ntorki u pojedinom entitetu (naziv, tip, opis, pripadnost entitetu, referencijski integritet...), pri čemu se svaki atribut opisuje jednim zapisom u metapodacima

Struktura podataka jednostavno se preslikava u repozitorij metapodataka, zahvaljujući činjenici da relacijske baze podataka sadrže pohranjene metapodatke o strukturi entiteta. Na taj način, bez većih dodatnih transformacija, moguće je stvoriti opis strukturiranih zapisa u repozitoriju metapodataka.

Gledajući sa strane prostorno-vremenskih podataka, vremenska domena pojedine ntorke mora biti unaprijed poznata, odnosno eksplicitno definirana u izvoru podataka. Podatak o pripadnoj vremenskoj oznaci pojedinog zapisa očekuje se prilikom svake dostave, bez potrebe za implicitnom derivacijom podatka na temelju njegovog okruženja.

Dostava strukturiranih podataka odvija se isključivo na ulazna sučelja koja podržavaju prihvat strukturiranih podataka. Kroz podatkovne cjevovode obavljaju se procesi integracije i gutanja strukturiranih podataka korištenjem prethodno razvijenih ugrađenih procedura ili ETL procesa. Važno je istaknuti kako se ETL procesi implementiraju ukoliko je skladišta podataka dio jezera podataka, odnosno kada je u sustavu potrebno osigurati skladištenje podataka prema odgovarajućim postulatima navedenima u [47].

Gutanje strukturiranih podataka gleda se kao podskup integracije podataka, budući da se radi o serijskoj obradi podataka. Integracija podataka korištenjem ETL procesa uključuje provedbu transformacija, što je u suprotnosti s gutanjem podataka te stoga procesi gutanja podataka pripadaju isključivo konačnoj pohrani u postrelacijske baze. Pritom se pod pojmom

postrelacijskih baza misli na baze koje nisu dimenzijski modeliranje principima iz skladištenja podataka. Implementacijski, gutanje podataka i integracija podataka mogu biti izvedeni korištenjem istovjetnih alata, pri čemu gutanje podataka neće obavljati operacije transformacija.

Strukturirani podaci mogu se preslikavati u polustrukturirane podatke korištenjem automatiziranih procesa, temeljenih na metapodacima. Odnosi između entiteta poznati su kroz strane ključeve, odnosno veze u repozitoriju metapodataka. Takvi se odnosi preslikavaju u ugniježđene sheme, pri čemu svaka ugniježđena shema odgovara jednom entitetu. Kardinalnosti oblika jedan-na-više preslikavaju se u shemu kao lista instanci zapisa. Na razini atributa, iz strukturiranih zapisa svaki se atribut preslikava identično u shemu polustrukturiranog zapisa, zadržavajući izvorni tip podataka.

Na strukturirane podatke primjenjuju se relacijski operatori oblika *relacija-u-relaciju* (*eng. relation-to-relation*). Kako su navedeni operatori dobro poznati i ugrađeni u sustave za upravljanje relacijskim bazama podataka (SUBP), za njihovo izvršavanje nisu potrebne dorade u kontekstu jezera podataka. Za potporu obradi prostorno-vremenskih tokova podataka, SUBP trebaju u podržanom skupu operatora sadržavati operatore nad vremenski nepromjenjivim tipovima podataka te operatore nad vremenski promjenjivim tipovima podataka.

Operatori oblika *relacija-u-dokument* (*eng. relation-to-document*) osiguravaju preslikavanje relacija, odnosno strukturiranih zapisa, u dokumente, odnosno polustrukturirane zapise. Navedeni operatori opisani su u [32] te se očekuje podržavanje osnovnog skupa operacija te operacija za potporu prostornim i vremenskim tipovima podataka.

### 1.5.2. Polustrukturirani izvori podataka

U polustrukturiranim izvorima podataka očekuju se svi tipovi podataka sadržani u skupu podržanih tipova podataka. Svi izvori podataka temeljeni na polustrukturiranim zapisima imaju prethodno dobro poznatu strukturu definiranu putem odgovarajućih sučelja i opisanu metapodacima.

Kao potpora prostorno-vremenskim podacima, u ovom se slučaju očekuje da je vremenska dimenzija podataka definirana eksplicitno u samom izvoru podataka. To ujedno podrazumijeva kako nema potrebe za implicitnim računanjem njezine vrijednosti tijekom obrade.

U polustrukturiranim izvorima podataka, unutar entiteta očekuju se hijerarhije podataka iskazane kao ugniježđeni dokumenti. Preslikavanje hijerarhija u entitetima u relacije obavlja se

pomoću unaprijed definiranih procedura, temeljenih na operatorima *document-to-relation*, koje su pohranjene u opisu pojedinog entiteta u repozitoriju metapodataka. Korištenjem procedura definiranih u metapodacima omogućena je automatizacija procesa preslikavanja.

Polustrukturirane izvore podataka opisujemo metapodacima do razine pojedinog atributa sadržanog u shemi izvora podataka. Prvenstveno, metapodacima je opisan sam izvor podataka, nazivom i opisom te tekstualnim zapisom sheme. Entiteti se opisuju u cijelom opsegu hijerarhije, za svaki ugniježđeni entitet unutar dokumenta zasebno. Pojedini entitet opisuje se nazivom i opisom, pripadnošću pojedinom izvoru podataka te funkcijom za preslikavanje u pripadni konačni oblik korišten u podatkovnim cjevovodima. Uz navedenu funkciju, potrebno je zapisati i podatke o entitetu koji odgovara konačnom obliku u perzistentnoj memoriji. Konačno, metapodacima su opisani atributi koji su, pored naziva, opisa i tipa podataka, pridijeljeni pojedinom entitetu. Kako pojedini atribut ujedno može biti ugniježđeni entitet, potrebno je osigurati da se taj podatak može iskazati metapodacima, na način da su atribut i njegov pripadni ugniježđeni entitet povezani.

U polustrukturiranim izvorima podataka, shema podataka je definirana u maksimalnom opsegu podataka koji se mogu očekivati. No, svaki dostavljeni zapis ne mora sadržavati sve atribute iz sheme, što se prilikom definiranja procedura preslikavanja u strukturirane podatke mora uzeti u obzir. Također, zahvaljujući činjenici da je shema podataka za pojedini izvor unaprijed definirana, moguće je automatikom definirati sve metapodatke, na sličan način kao i u slučaju strukturiranih izvora podataka.

Podaci iz polustrukturiranih izvora podataka trebaju se pohraniti u svom prirodnom obliku kroz procese gutanja podataka. No, istovremeno je dostavljene podatke moguće uključiti u podatkovne cjevovode koji obavljaju procese integracije podataka. Pri tom se u procese implementiraju operacije oblika *dokument-u-relaciju* (eng. *document-to-relation*), što povećava kompleksnost integracije podataka u odnosu na tradicionalnu integraciju temeljenu isključivo na relacijskim podacima. Navedene operacije odgovaraju objektno-relacijskim preslikavanju (eng. *object-relational mapping, ORM*), metodologiji poznatoj iz objektno-orijentiranog programiranja.

U procesu integracije podataka također se mogu očekivati implementacije operatora oblika *dokument-u-dokument (eng. document-to-document)*. Njihovim korištenjem kreiraju se u podatkovnim cjevovodima novi dokumenti koje se potom koriste u daljnjim obradama te po potrebi i zapisuju u perzistentnu memoriju. Navedeni operatori ponašaju se kao i relacijski

operatori i podržavaju barem podskup poznatih relacijskih operatora. U svrhu obrade podataka, potrebni su barem operatori selekcije, projekcije i spajanja. Ostali operatori, poput operatora grupiranja i agregacije, iako nisu nužni za potporu osnovnoj obradi podataka, otvaraju dodatne mogućnosti u analizi i obradi podataka.

Kada se obavljaju operacije oblika *dokument-u-relaciju* i u podatkovnom cjevovodu rezultiraju stvaranjem jedne ili više relacija, može se očekivati da rezultat bude spljoštena (eng. *flattened*) ntorka u 1NF. U slučajevima kad je konačni zapis u 3NF, potrebno je osigurati pomoću metapodataka normalizaciju spljoštene ntorke do 3NF.

### 1.5.3. Nestrukturirani izvori podataka

Nestrukturirane izvore podataka odlikuje činjenica da podatci nemaju konačnu strukturu u trenutku dostave podataka, već se ona formira nakon što su podaci prihvaćeni ili pohranjeni u perzistentnu memoriju. Nestrukturirani podaci svoju konačnu strukturu dobivaju obavljanjem pripadnih operacija preslikavanja, kojima prelaze u polustrukturirane ili strukturirane podatke.

U nestrukturiranim prostorno-vremenskim podacima očekuju se prostorna i vremenska oznaka, odnosno zapisi u prostornoj i vremenskoj domeni. U slučaju da se podaci dostavljaju kao jedan zapis u tipu *blob*, prostorna i vremenska oznaka stvaraju se prilikom operacije preslikavanja u odgovarajućim podatkovnim cjevovodima.

Potrebno je ujedno naglasiti da, iako podaci dolaze iz nestrukturiranih izvora, ne znači da ti podaci nemaju pripadnu strukturu. Razlika u odnosu na strukturirane i polustrukturirane podatke jest da je njihova struktura izuzetno varijabilna te se ne može jednoznačno odrediti sama struktura.

Nestrukturirani izvori podataka opisani su minimalno metapodacima o nazivu i opisu samog izvora, a moguće ih je opisati i dodatnim metapodacima ovisno o potrebi. Za svaki nestrukturirani izvor podataka očekuju se metapodaci o entitetu: naziv, opis i pripadni skup funkcija za preslikavanje u polustrukturirane i strukturirane oblike. Jedan entitet može imati više pridijeljenih funkcija za preslikavanje u konačne oblike. Čak i kada entiteti iz više izvora podataka imaju identičnu strukturu i funkcije za preslikavanje, moraju biti iskazani odvojenim zapisima metapodataka. Konačno, opisuju se i metapodacima o atributima (svaki entitet koji pripada nestrukturiranom izvoru podataka mora sadržavati barem jedan atribut). Valja istaknuti kako funkcije preslikavanja ne moraju nužno koristiti sve atribute iz izvorišnog entiteta.

Nestrukturirani podaci jednostavnije se preslikavaju u polustrukturirane nego u strukturirane zapise. Pri preslikavanju u strukturirane zapise, potrebno je obaviti preslikavanje u polustrukturirani oblik, a potom se korištenjem objektno-relacijskog preslikavanja dobiva strukturirani zapis.

Podaci iz nestrukturiranih izvora podataka u podatkovnim cjevovodima mogu se integrirati s polustrukturiranim i strukturiranim podacima korištenjem odgovarajućih procesa. Integracija nestrukturiranih podataka može biti performansno zahtjevan proces, stoga je kod nestrukturiranih podataka češće prisutno gutanje podataka uz naknadno obavljanje operacija preslikavanja i procesa integracije.

## 1.6. Sistematizacija tokova podataka prema strukturi podataka

Tokovi podataka, u kontekstu izvora podataka, odlikuju se neprekidnom dostavom podataka, što ih distancira od izvora podataka kod kojih su podaci konzistentni i uvijek dostupni. Tokovi podataka podijeljeni su prema strukturi podataka u strukturirane, polustrukturirane i nestrukturirane.

### 1.6.1. Strukturirani tokovi podataka

U strukturiranim tokovima podataka očekuju se svi tipovi podataka sadržani u skupu podržanih tipova podataka. Struktura podataka prethodno je opisana na strani samog toka podataka te se jednostavno iskazuje metapodacima.

U strukturiranim tokovima podataka vremenska dimenzija pojedine dostavljene ntorke može se definirati eksplicitno prilikom nastajanja podataka ili implicitno na temelju vremenske oznake. U slučaju dviju ili više različitih vrijednosti vremenskih oznaka u podacima na temelju kojih se treba odrediti pripadna vremenska oznaka, pri definiranju metapodataka potrebno je označiti koji atribut je primaran pri određivanju vrijednosti.

U predloženom modelu unificiranog sloja pristupa podacima, strukturirani tokovi podataka spajanju se na sučelja za strukturirane podatke koja imaju definirane pripadne duljine prozora i omogućen stvarnovremenski prihvat podataka.

Ovisno o volumenu i brzini pristizanja podataka, podaci iz strukturiranih tokova podataka mogu se integrirati s ostalim podacima pomoću definiranih procedura ili integracijskih procesa. Kada volumen ili brzina pristizanja predstavljaju performansne izazove za procese integracije, podatkovni cjevovodi moraju omogućiti gutanje podataka u jezero podataka te pristup privremeno pohranjenim podacima kroz Lambda arhitekturu. U svrhu integracije, pohranjeni se podaci iz strukturiranih tokova podataka dohvaćaju iz perzistentne memorije kao serije podataka.

Sučelja za strukturirane podatke na koja pristižu podaci iz strukturiranih tokova podataka trebaju podržavati upitni jezik CQL i operatore povezane s njim:

- Operatori nad vremenski nepromjenjivim tipovima podataka
- Operatori nad vremenski promjenjivim tipovima podataka
- Operatori prozora

U radu sa strukturiranim tokovima podataka dominantni su operatori oblika *stream-to-relation*.

### 1.6.2. Polustrukturirani tokovi podataka

Polustrukturirani tokovi podataka temelje se na korištenju shema polustrukturiranih podataka. Kao i slučaju strukturiranih tokova podataka, vremensku dimenziju podataka može se definirati implicitno prilikom stvaranja podataka i eksplicitno u trenutku dohvata podataka. Eksplicitna vremenska oznaka očekuje se među dostavljenim podacima kao atribut sheme. U slučaju izostanka eksplicitno definirane vremenske oznake, nužno je implicitno definirati vrijednost prilikom dohvata podataka na sučeljima za polustrukturirane tokove podataka. Referentni atribut za vremensku dimenziju pojedinog zapisa definira se u repozitoriju metapodataka.

Polustrukturirani izvori tokova podataka, kao podskup polustrukturiranih izvora podataka, podržavaju rad sa svim tipovima podataka sadržanima u skupu podržanih tipova podataka.

Specifičnost polustrukturiranih tokova podataka je mogućnost jednostavnijeg spajanja heterogenih shema unutar podatkovnih cjevovoda u odnosu na strukturirane podatke. Jednostavnije spajanje posljedica je fleksibilne, proširive sheme. Više spojenih polustrukturiranih tokova podataka stvara novu shemu koja obuhvaća sve atribute, uključujući i dijeljene atribute iz pripadnih tokova podataka uz preslikavanje njihovih vrijednosti.

Ključ spajanja tokova podataka može biti odabran između vremenskih i prostornih dimenzija te prirodnih ključeva samih shema. Spajanje se može obaviti u novu shemu koja nastaje kao unija svih shema ili shemu dijeljenim atributima i ugniježđenim dokumentima. U slučaju spajanja u jednu shemu, ona se definira kao unija svih atributa tokova podataka koji se spajaju te se kao nova shema zapisuje u repozitorij metapodataka. S druge strane, kada se stvara nova shema s ugniježđenim dokumentima, dijeljeni atributi s pripadnim zapisima zapisuju se kao atributi nove sheme, a preostali atributi i zapisi odvajaju se u zasebne ugniježđene sheme. Nova shema se također definira u repozitoriju metapodataka, no zahtjeva kompleksniju definiciju zbog drugačije kardinalnosti.

U slučaju tokova polustrukturiranih podataka, dominantni su procesi gutanja podataka, budući da se podaci čak i prilikom spajanja tokova podataka ne mijenjaju, već se samo stvara nova shema. Spajanje tokova podataka obavlja se unutar podatkovnih cjevovoda, s mogućnošću pohrane međurezultata operacija spajanja u perzistentnu memoriju.

U radu s tokovima polustrukturiranih podataka prisutni su operatori tipa *stream-to-document*:

- Operatori nad vremenski nepromjenjivim tipovima podataka
- Operatori nad vremenski promjenjivim tipovima podataka
- Operatori prozora

### 1.6.3. Nestrukturirani tokovi podataka

U nestrukturirane tokove podataka ubrajamo sve tokove podataka čiji sadržaj nema prethodno definiranu strukturu te podatke koji imaju strukturu, no ona ne odgovara shemama relacija ili polustrukturiranim shemama.

Kada se radi o potpuno nestrukturiranim podacima, potrebno je obraditi njihove značajke iz metapodataka kako bi se definirale vremenska i prostorna domena zapisa. Pojedini tokovi podataka mogu sadržavati zapise koji posjeduju vlastite metapodatke u obliku definiranih i standardiziranih normi (primjerice, slikovni zapisi). Za takve zapise, metapodaci se mogu djelomično automatski dohvatiti i pohraniti u repozitorij metapodataka. Ukoliko ne postoje u metapodacima, potrebno je eksplicitno definirati atribute koji sadrže prostornu i vremensku komponentu. Vremenski i prostorni podaci se potom pohranjuju bilo na temelju eksplicitne vrijednosti, bilo kroz implicitnu definiciju iz samog toka podataka.

Vremenska dimenzija može se prvotno definirati implicitno, preuzimanjem vremena pojave u toku podataka. Implicitna vrijednost može se zamijeniti eksplicitnom vrijednosti po završetku pohrane i obrade podataka

Prostorna dimenzija u slučaju stacionarnog objekta može se eksplicitno definirati na temelju poznate lokacije (fiksno postavljene prostorne komponente objekta), dok se kod pokretnih objekata očekuje dostavljanje vrijednosti atributa prostorne dimenzije.

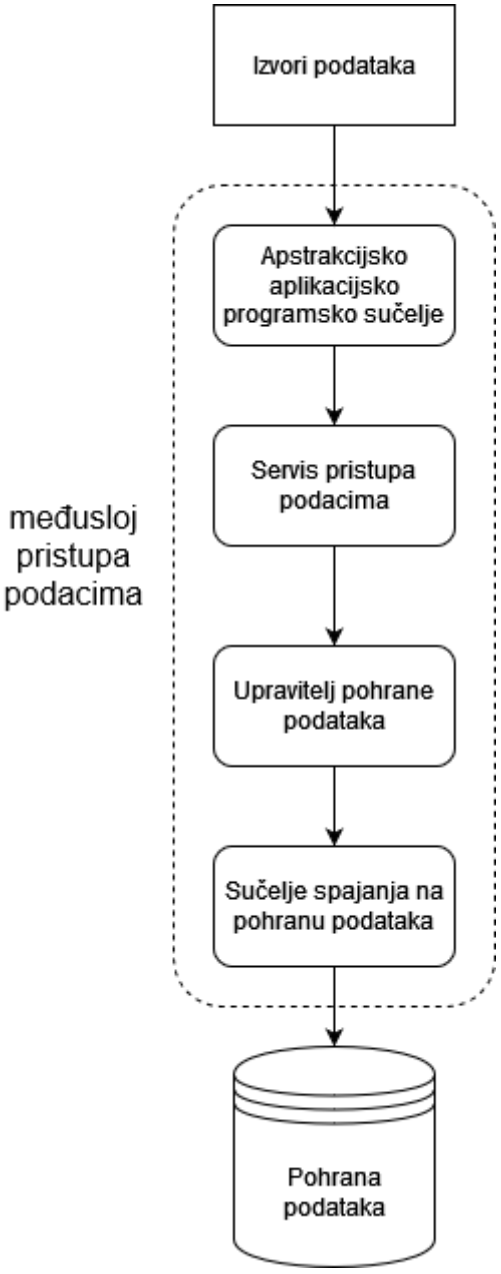
Spajanje objekata obavlja se u pravilu nakon gutanja podataka i pohrane u jezero podataka, no uz prilagodbu cjevovoda moguće je jednostavnija spajanja obaviti neposredno nakon gutanja podataka. Tako se u jezero podataka mogu zapisati već spojeni podaci.

Mogućnost spajanja u cjevovodu ovisi o kompleksnosti operacija spajanja – stvaranje redova čekanja (eng. *queue*) u cjevovodu nakon gutanja podataka nije prihvatljivo kao posljedica operacije. Spajanja se arhitekturalno postavljaju u cjevovod ovisno o kompleksnosti operacije koja proizlazi iz stupnja heterogenosti zapisa u tokovima podataka, volumenu podataka u tokovima podataka i brzini pristizanja tokova podataka koji se spajaju

Ako spajanje nije moguće obaviti u vremenu koje ne stvara redove čekanja, spajanje podataka mora se izvršiti nakon zapisivanja u perzistentnu memoriju.

## 1.7. Model međusloja pristupa podacima

U radu [46] prikazan je tipični model međusloja pristupa podacima korišten za rad s NoSQL bazama podataka. NoSQL baze podataka specifične su zbog značajke posjedovanja aplikacijskog programskog sučelja za komunikaciju s izvorima i odredištima podataka, poput programskih aplikacija. Predloženi model razložen je na komponente koje tvore zajednički podatkovni cjevovod, kao što je prikazano na slici 8. Komponente međusloja pristupa podacima na slici omeđene su iscrtanom linijom.




Slika 8. Model međusloja pristupa podacima

Apstraktno aplikacijsko programsko sučelje zaduženo je za komunikaciju prema vanjskim aplikacijama. U njemu je definiran model podataka, dostupan vanjskim aplikacijama koje pomoću njega definiraju svoje interne modele podataka. Servis pristupa podacima prilagođava dolazne podatke u međuoblik, pogodan za daljnju obradu i pohranu u konačno odredište. Pored prilagodbe podataka, u servisu pristupa podacima obavljaju se specifične zadaće međusloja, poput privremene pohrane podataka. Upravitelj pohrane podataka preuzima podatke u prilagođenom međuobliku i konačno ih pretvara u model pogodan za komunikaciju putem sučelja spajanja na pohranu podataka. Sučelje spajanja na pohranu podataka predstavlja uniformno sučelje s mogućnošću spajanja različitih upravljačkih programa (eng. *driver*). Upravljački programi prilagođeni su za svaki pojedini oblik pohrane podataka te interno sadržavaju sučelja specifična za pojedinu instancu pohrane podataka. Time uniformna sučelja mogu podržati spajanje više različitih oblika pohrane podataka.

Međusloj pristupa podacima, uvođenjem podatkovnih cjevovoda, stvara određeni međuslojni trošak koji ovisi o kompleksnosti komponenata, odnosno servisa pristupa podacima i upravitelja pohrane podataka.

Model međusloja predložen u [46] zadovoljava potrebe rada s NoSQL bazama podataka, što uključuje i potporu polustrukturiranim podacima s poznatom shemom. Uzevši to u obzir, predloženi model poslužio je kao osnova za prilagodbu u unificirani sloj pristupa podacima u jezeru podataka.

## 1.8. Pohrana podataka u jezerima podataka

### 1.8.1. Pohrana strukturiranih podataka

Strukturirani podaci najčešći su oblik pohrane u transakcijskim sustavima, no nisu ograničeni samo na njih. Glavna odlika strukturiranih podataka je fiksna struktura, definirana u sustavu za pohranu podataka.

Za svaki skup podataka kojeg smatramo strukturiranim potrebno je prethodno definirati njegovu strukturu ntorke u obliku entiteta u repozitoriju metapodataka. U okviru predloženog modela jezera podataka, zapis je potrebno definirati u postrelacijskoj bazi podataka i u repozitoriju metapodataka kako bi struktura bila jedinstveno definirana na razini cijelog modela.

Svaka promjena u strukturi zapisa, analogno prethodno opisanom postupku definiranja strukture, mora se obaviti u postrelacijskoj bazi i u repozitoriju metapodataka. Gledano s tehničke strane, postupak propagiranja promjene iz postrelacijske baze u repozitorij metapodataka moguće je automazitirati kroz definiranje odgovarajućih okidača na strani postrelacijske baze i povezanih operacija u repozitoriju metapodataka. U repozitoriju metapodataka, pojedini entitet iz postrelacijske baze podataka opisuje se kroz instancu klase objekt uz pripadni skup klasa atributa.

Ključna značajka u odabiru postrelacijske baze podataka za model jest potpora za prostorno-vremenske tipove podataka i operacije, opisane u radu [32]. Dok većina sustava za upravljanje bazama podataka podržava rad s prostornim podacima, prostorno-vremenski tipovi podataka zahtijevaju određene dorade prilikom implementacije. Izazovima proširenja postrelacijskih baza podataka u svrhu potpore prostorno-vremenskim tipovima i operacijama posvećen je dio istraživanja u [32]. U istom radu opisana je i potpora za rad s prostorno-vremenskim tokovima podataka u postrelacijskim bazama podataka.

Pri definiranju entiteta unutar postrelacijske baze podataka ključno je definirati ispravni primarni ključ. U slučaju entiteta koji sadrže prostorno-vremenske tipove podataka, dio primarnog ključa svakako nužno moraju biti vremenska oznaka zapisa te prostorna značajka zapisa. Na taj način osigurava se mogućnost praćenja kretanja prostornog zapisa kroz vremenski tok, neovisno o njegovom prostornom obliku.

Kada se povuče paralela s Lambda arhitekturom, strukturirani podaci odgovaraju sloju serija podataka. Postrelacijske baze pružaju mogućnost gotovo stvarnovremenske obrade i pohrane podataka, pogotovo u slučaju kad nema potrebe za transformacijama u obradi podataka. No

jedna njihova glavna odlika – konzistentnost podataka – usporava pohranu u kompleksnim modelima baza podataka. Istovremeno, konzistentnost podataka daje postrelacijskoj bazi mogućnost da se promatra kao jedinstveni izvor točnih podataka.

Modeliranje postrelacijske baze podataka za potporu prostorno-vremenskim tokovima podataka vrlo je blisko modeliranju skladišta podataka. Skladišta podataka definiramo kao subjektno orijentirane, integrirane, vremenski ovisne i nepromjenjive skupove podataka sa svrhom potpore procesima odlučivanja [47]. Većinu navedenih značajki skladišta podataka možemo primijeniti i na prostorno-vremenske postrelacijske baze podataka:

- Svaki prostorno-vremenski podatak sadrži vremensku komponentu. Zapis koji se pohranjuje u postrelacijsku bazu podataka nužno mora imati različitu oznaku vremena, iz čega proizlazi da su ti podaci vremenski zavisni.
- Jednom dostavljen podatak iz toka podataka ne može se promijeniti. Svaka promjena u zapisima dolazi nužno s novom vremenskom oznakom i promatra se kao novi zapis
- Podatke iz više tokova podataka moguće je integrirati u jednu relaciju u postrelacijskoj bazi podataka

Uzevši u obzir analogiju skladišta podataka i prostorno-vremenske postrelacijske baze podataka na temelju spomenutih značajki te činjenicu se skladišta podataka smatraju jedinstvenim izvorom istine za podatke, može se smatrati da su i prostorno-vremenske postrelacijske baze podataka također jedinstveni izvor točnih podataka.

Najveći nedostatak pohrane strukturiranih podataka svakako su u većoj mjeri brzina procesa pohrane podataka, dok brzina procesa dohvata podataka u manjoj mjeri utječe na potporu stvarnovremenskim operacijama. Najčešći uzrok usporavanju svakako je potreba za spajanjem više entiteta unutar jednog upita, čemu najviše pridonosi zahtjevnost same operacija spajanja. Kako bi se izbjegao negativni utjecaj na performanse, model postrelacijske baze podataka trebao bi biti denormaliziran i optimiziran za dohvat podataka.

Najveća prednost denormaliziranog modela postrelacijske baze podataka leži u činjenici da se ne očekuje ažuriranje samih podataka, odnosno da se jednom uneseni podatak neće promijeniti, dok će se podaci često dohvaćati. Denormalizacijom podataka ujedno se zadaća održavanja integriteta podataka prenosi na sloj zadužen za integraciju podataka i rasterećuje postrelacijska bazu podataka.

### 1.8.2. Pohrana nestrukturiranih podataka

Jedna od glavnih odlika jezera podataka je brza pohrana heterogenih podataka, neovisno o njihovoj brzini stvaranja ili strukturi. Za razliku od postrelacijskih baza podataka u kojima su podaci strukturirani i NoSQL baza podataka s uglavnom polustrukturiranim podacima, u jezerima podataka podaci se mogu pohraniti bez da se prethodno njihova struktura mora promijeniti nekom transformacijom. Najčešće se pritom radi o pohrani nestrukturiranih podataka.

U kontekstu pohrane podataka, nestrukturiranim podacima smatramo sve podatke koji nemaju formalno definiranu konačnu strukturu u trenutku prihvata

Sama definicija nestrukturiranih podataka ostavlja i dalje mogućnost da podaci imaju određenu generalnu strukturu, no ona se može mijenjati prilikom svakog novog dostavljanja podataka. Pritom treba nestrukturirane podatke razlučiti od polustrukturiranih podataka, koji mogu biti dostavljani s određenim, unaprijed poznatim dijelovima strukture, no u slučaju polustrukturiranih podataka ona je formalno definirana najvećim skupom značajki koje je opisuju.

Postoji niz razloga koji ukazuju na važnost pohrane nestrukturiranih podatka u izvornom obliku. Prvenstveno, nestrukturirani podaci u sebi sadrže informacije koje su teško ili nikako dostupne klasičnim pristupima prikupljanja informacija (kao što su, primjerice, izvještajni alati). U svrhu dohvata informacija iz nestrukturiranih podataka koriste se procesi dubinske analize podataka (*eng. data mining*) te alati i procesi napredne analitike. Nadalje, obrada u svrhu strukturiranja i integracije podataka, koristeći postojeće pristupe obrade strukturiranih podataka, sa sobom nosi rizik značajnog opterećenja sustava i gubitka mogućnosti stvarnovremenskog pristupa podacima.

Kako se radi o velikom volumenu heterogenih podataka, proces njihove obrade zahtjevan je čak i za raspodijeljene sustave. Iz tog razloga, kako bi podaci postali dostupni u sustavu za pohranu nestrukturiranih podataka, oni se pohranjuju u izvornom obliku, a transformacije se odvijaju u trenutku dohvata pohranjenih podataka. Na taj način podaci postaju dostupni za daljnju obradu s minimalnom latencijom, gotovo stvarnovremenski, dok su ujedno naknadno pohranjeni u trajnoj memoriji, odakle se po potrebi mogu obrađivati u serijama.

Pohrana nepromijenjenih heterogenih podataka odgovara sloju brzih podataka Lambda arhitekture. Kako u Lambda arhitekturi sloj brzih podataka pruža pristup samo najnovijim

podacima, potrebno je osigurati prijenos isteklih podataka iz brze trajne memorije u sporiju trajnu memoriju, odakle su i dalje dostupni na zahtjev. Pohranom u trajnu memoriju automatski se obavlja indeksiranje, nakon čega podaci postaju dostupni uslužnom sloju.

Razlog izbjegavanja indeksiranja u brzoj memoriji prvenstveno leži u namjeni brze memorije. U njoj se podaci mijenjaju velikom brzinom i održavanje bilo kakvih indeksa usporilo bi cijeli proces, stoga je prihvatljivije dohvatiti veći skup podataka i obaviti operaciju projekcije unutar alata za analizu podataka.

U okviru prostorno-vremenskih podataka, potrebno je uzeti u obzir da i u nestrukturiranim podacima postoji definirana prostorna i vremenska komponenta. Definiranjem podržanih prostorno-vremenskih tipova podataka, moguće je uvesti djelomičnu strukturu u nestrukturirane podatke, čime se ujedno i omogućuje indeksiranje samih podataka prostorno-vremenskim indeksima.

U svrhu efikasnijeg rada s prostorno-vremenskim nepromijenjenim podacima uvode se proširenja postojećih sustava za pohranu podataka. U programskom okviru Hadoop, primjerice, razvijena su proširenja SpatialHadoop [48] i njegova nadogradnja ST-Hadoop [49] u obliku proširenja MapReduce modela. Iako takve vrste proširenja nisu preduvjet za pohranu podataka, njihovim korištenjem povećava se efikasnost prostorno-vremenskih upita kroz uvođenje prostorno-vremenskih indeksa.

### 1.8.3. Pohrana polustrukturiranih podataka

Pohrana polustrukturiranih podataka u pravilu se temelji na NoSQL bazama podataka, iako nisu sve vrste NoSQL baza podataka pogodne za njihovu pohranu. U ovom radu i sistematizaciji fokus će biti isključivo na dvije vrste: graf bazama podataka i dokument bazama podataka. Također, izostavljeno je analiziranje baza podataka za pohranu XML polustrukturiranih zapisa, budući da su u praksi dokument baze podataka zastupljenije.

Graf baze podataka, u domeni prostorno-vremenskih podataka, imaju zadaću pohrane prostornih podataka zajedno s proizvoljnim brojem značajki. Svaki čvor u graf bazi odgovara prethodno definiranom prostornom tipu, što je ujedno i njegova osnovna značajka. U graf bazi čvorovi su označeni identifikatorima koji se generiraju interno u sustavu za upravljanje bazom podataka. Same graf baze mogu se pretraživati prema značajkama, no prava namjena im je stvaranje odnosa putem bridova među čvorovima koji ujedno i opisuju te odnose.

Uloga graf baze podataka u svrhu pohrane prostornih podataka je omogućavanje efikasne podloge za izvođenje algoritama temeljenih na odnosima između prostornih podataka. Jedan od primjera je pretraga najkraćeg puta u prostoru, pri čemu je bitan odnos susjedstva među prostornim zapisima kako bi se stvorila putanja između indirektno povezanih podataka.

Kako prostorno-vremenski podaci ne tvore vertikalne hijerarhije, u modelu graf baze podataka nije potrebno ostvarivanje više slojeva podataka. Podaci se mogu pohraniti u trodimenzionalnom prostoru kreiranjem čvorova čiji je atribut lokacija u prostoru, no sam model predstavlja se kao jedinstveni usmjereni graf. Važno je istaknuti i da je svaki smjer kretanja predstavljen kao zasebni brid koji označava mogućnost pomaka u određenom smjeru između dva čvora, odnosno prostorne točke.

Podaci u dokument bazi podataka pohranjeni su u polustrukturiranom obliku koji prati model razmjene objekata (eng. *Object Exchange Model - OEM*) i zapisani u JavaScript Object Notation obliku. U dokument bazi podataka prostorno-vremenski podaci pohranjuju se korištenjem GeoJSON standarda zapisa [50] [51]. GeoJSON standard zapisa temelji se na široko korištenom JavaScript Object Notation obliku zapisa uz proširenja prostornim tipovima podataka.

Vremenska komponenta zapisa u dokument bazi podataka pohranjuje se u tipu vremenske oznake. Glavna prednost korištenja tipa vremenske oznake je u potpori analitičkim aplikacijama kroz ugrađene funkcije za rad s vremenskim oznakama, što olakšava kreiranje agregacija i analizu vremenskih komponenti na unificiran način, neovisno o načinu pristupa podacima.

Dokumenti se modeliraju prema značajkama pojedinih izvora podataka u polustrukturiranim obliku. Pojedini dokument može biti ograničen na jedan izvor podataka, na skup izvora podataka ili na vremensku oznaku. Izbor granulacije dokumenta ovisi o volumenu i frekvenciji dolaska podataka. Više dokumenata grupira se u jednu kolekciju ovisno o njihovoj namjeni. Pojam kolekcije u polustrukturiranoj pohrani analogan je entitetu u strukturiranom obliku pohrane, dok je pojam dokumenta analogan ntorki.

Pohrana podataka u polustrukturiranom obliku, u odnosu na strukturirani oblik, omogućava značajno bržu pohranu izbjegavanjem zahtjevnih operacija transformacije u svrhu integracije podataka. Također, u slučaju hijerarhijski organiziranih podataka, model polustrukturirane pohrane je superiorniji strukturiranom iz razloga što se time izbjegava operacija spajanja prilikom dohvata podataka.

# 2. Vezana istraživanja

Jezera podataka, kao brzorastuća tehnologija u proteklom desetljeću, nisu u jednakoj mjeri praćena kroz formalna istraživanja i definicije. Ujedno, zbog diskrepancije između implementacija i nedovoljnog stupnja zrelosti, formalizacija naprednijih modela jezera podataka tek je naknadno uspostavljena. Unutar jezera podataka, uz formalno predložene arhitekture, dodatno se sagledava način obrade podataka kao kombinacije obrada serija podataka i tokova podataka. Takav pristup potreban je u svrhu potpore stvarnovremenskoj dostupnosti podataka koji pristižu velikom brzinom.

Procesi upravljanja podacima imaju važnu ulogu u kvalitetnoj implementaciji analitičkih sustava. Njihovom definicijom u cijelom poslovnom okruženju, ali ponajviše u skladištima podataka i jezerima podataka, značajno se doprinosi zrelosti sustava i uspostavljanju odgovornosti u korištenju podataka. Kroz procese upravljanja podacima ujedno se definira korištenje metapodataka u jezerima podataka, što je jedan od preduvjeta za uspješnost implementacije i konačnog prihvaćanja samog jezera podataka. U ovom poglavlju prikazano je nekoliko različitih pristupa stvaranju modela metapodataka za potporu jezerima podataka koji dijele dio značajka, dok su najveće razlike u pristupu izradi arhitekture modela.

Iako prostorno-vremenski tokovi podataka nisu novost, razvojem tehnologije, prvenstveno pokretnih senzora i interneta stvari, dobili su dodatno na značaju. Iako se njihova struktura podataka ujedno i mijenjala od strukturiranih prema nestrukturiranima, osnova prostorno-vremenskih tipova podataka i operacija nad njima dobro je definirana. Ujedno, takvi definirani sustavi tipova podataka postaju i osnova za potporu tokovima podataka u jezerima podataka.

## 2.1. Istraživanja u domeni jezera podataka

### 2.1.1. Definicija jezera podataka

U [4] dan je pregled postojećih istraživanja u domeni jezera podataka, s posebno istaknutim smjerovima fokusa istraživanja. Kao primjeri, navode se koncepti i definicije jezera podataka, kao istraživanje na višoj razini, te istraživanje tehnologija potrebnih za implementaciju jezera podataka. Isti rad detaljno je prikazao postojeće definicije jezera podataka, od prvog pojavljivanja pojma [5] sve do novijih definicija. Kao jedna od definicija navodi se centralni repozitorij u kojem su podaci u svim dostupnim oblicima pohranjeni bez striktne sheme u svrhu budućih analiza. Uz navedenu definiciju veže se i pristup kreiranja sheme prilikom čitanja, odnosno naknadnog vezivanja podataka [6], koji je uz veliku raznolikost podataka jedna od glavnih karakteristika jezera podataka.

Jezera podataka u [7] definirana su kao logički pogled svih izvora i skupova podataka u njihovom nepromijenjenom obliku, dostupno za dohvat znanja iz podataka. Uz to, dan je skup značajki koje jezero podataka mora zadovoljavati

- kvaliteta podataka osigurana je kroz skup metapodataka
- jezero podataka kontrolirano je pravilima, alatima i procesima kako bi upravljanje podacima bilo zajamčeno
- korištenje jezera podataka ograničeno je za statističare i podatkovne znanstvenike kako bi sigurnost, privatnost i usklađenost podataka bili zajamčeni
- jezero podataka pristupa svim tipovima podataka
- jezero podataka ima logičku i fizičku organizaciju

Na temelju prethodnih istraživanja [1], u [8] predloženo je proširenje nužnih značajki sa značajkom skalabilnosti kako bi definicija uključivala i podršku velikim podacima. Također, značajku ograničavanja korištenja donekle se redefinira, na način da se definira mogućnost korištenja svim korisnicima, ali uz naglašavanje korištenja od strane podatkovnih specijalista u svrhu dohvata znanja iz podataka.

### 2.1.2. Upravljanje podacima u jezerima podataka

Koncept upravljanja podataka (eng. *Data Governance*) nije striktno ograničen na jezera podataka, no njihovom pojavom, kao i neprekidnim uključivanjem novih izvora podataka, došao je posebno do izražaja. U [9] kroz istraživanje o prethodnim programskim okvirima upravljanja podacima, analiziran je prijedlog definicije upravljanja podacima na temelju ranijih

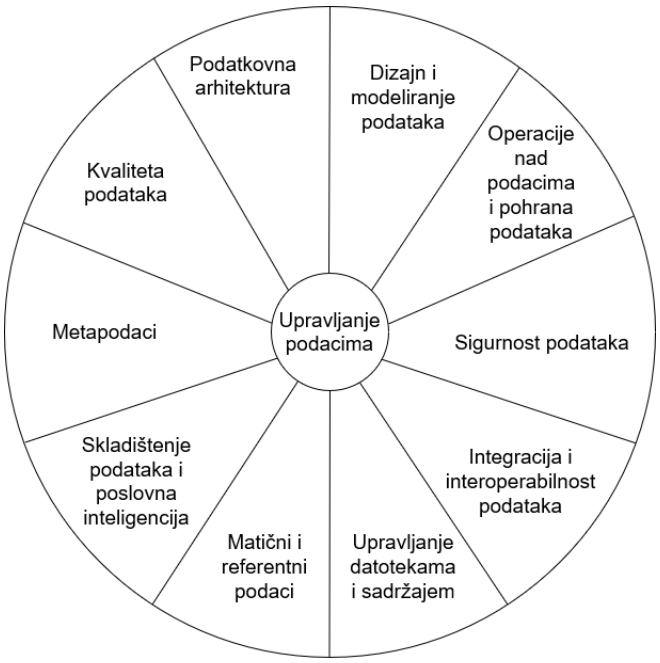
radova, pri čemu se najboljom pokazala definicija iz [10]. Navedeni rad definira upravljanje podacima kao programski okvir na razini cijelog poslovnog okruženja za dodjeljivanja prava i zadaća vezanih uz donošenje odluka sa svrhom adekvatnog rukovanja podacima kao imovinom poduzeća. DAMA (Data Management Association), krovna udruga koja okuplja stručnjake za upravljanje podacima, definirala je upravljanje podacima u [11] [12] kao izvršavanje ovlasti, kontrole i zajedničkog donošenja odluka (koje se sastoji od planiranja, praćenja i provedbe) nad upravljanjem podatkovnom imovinom (eng. *data assets*). U definicijama iz [10] i [12] vidljive su glavne odrednice koncepta: potpora donošenju odluka kroz uvođenje nadzora i kontrole nad podatkovnom imovinom.

Tradicionalno upravljanje podataka zadovoljavalo je potrebe [13] organizacija s računalnom potporom temeljenom na strukturiranim podacima pohranjenima u lokalnim podatkovnim centrima. Pritom je DAMA u [11] definirala koncept s devet područja znanja, u kojima su propisana ključna područja i standardi u upravljanju podacima. U navedena područja znanja spadaju:

- Upravljanje podatkovnom arhitekturom (eng. *Data Architecture Management*)
- Razvoj podataka (eng. *Data Development*)
- Upravljanje operacijama u bazama podataka (eng. *Database Operations Management*)
- Upravljanje sigurnosti podataka (eng. *Data Security Management*)
- Upravljanje referentnim i glavnim podacima (eng. *Reference & Master Data Management*)
- Upravljanje skladištima podataka i poslovnom inteligencijom (eng. *Data Warehousing & Business Intelligence Management*)
- Upravljanje dokumentima i sadržajem (eng. *Document & Content Management*)
- Upravljanje metapodacima (eng. *Metadata Management*)
- Upravljanje kvalitetom podataka (eng. *Data Quality Management*)

Veliki broj nestrukturiranih podataka koji zahtijevaju brzu obradu i pohranu doveo je do potrebe za novim pristupom upravljanju podacima. Stoga je prethodno definirani koncept djelomično izmijenjen kroz redefiniranje postojećih i dodavanje novih ključnih područja. Tako je područje razvoja podataka definirano kao modeliranje i dizajn podataka (eng. *Data Modeling & Design*), a dodano je novo područje integracije i interoperabilnosti podataka (eng. *Data Integration & Interoperability*).

Konačno, područje znanja upravljanja podacima je krovno područje koje objedinjuje sva navedena područja znanja u zajednički koncept, kao što je prikazano na slici 9:




Slika 9. DAMA područja znanja upravljanja podacima (preuzeto iz [11])

Izostanak definiranja dijela procesa upravljanja podacima nad jezerima podataka može dovesti do močvare podataka (eng. *data swamp*) [14] i posljedičnog gubitka povjerenja u sustav i njegovu korist. U radu [15] dan je primjer rješavanja izazova močvare podataka kroz implementaciju programskog okvira temeljenog na sličnosti podataka. Predloženi koncept prati tri kategorije značajki (značajke podataka, podrijetlo podataka i trenutne karakteristike) s ukupno deset značajki sličnosti pomoću kojih je, primjerice, moguće automatikom generirati metapodatke.

### 2.1.3. Lambda i Kappa arhitektura

Lambda i Kappa arhitekture često se uzimaju kao osnova gotovo stvarnovremenske obrade podataka unutar jezera podataka. Koncept Lambda arhitekture svoj temelj nalazi u radu [16], a u široku primjenu ulazi povećanjem volumena i brzine podataka, koji su doveli do nužnosti rada s velikim podacima. Postojeći alati za integraciju podataka, zbog kompleksnosti i latencije, prestali su biti dostatni za potporu odlučivanju u poslovnom okruženju koje zahitjeva što svježije podatke, što je u svom osvrtu u [17] iznio autor.

U [19] opširno je opisana implementacija sloja serija podataka Lambda arhitekture korištenjem tehnologija velikih podataka. Za prihvat podataka iz izvora zadužen je Apache

Kafka programski okvir, koji podatke distribuira prema slojevima za obradu podataka. Za sloj serija podataka korištena je Apache Hadoop platforma, koja uz pomoć MapReduce paradigme ima mogućnost paralelne obrade podataka i njihove pohrane u HDFS datotečni sustav. Sličan pristup naveden je i u [20].

Stvarnovremensko učitavanje i obrada podataka, budući da se obrađuje manji volumen podataka, omogućavaju korištenje analitičkih podataka uz značajno manju latenciju. Stoga se u sloju brzih podataka podaci ne pohranjuju trajno, već do pohrane u perzistentnu memoriju. Tako se u [18] navodi kako se trenutak brisanja starih podataka određuje završetkom obrade u sloju serija podataka, čime se izbjegava eventualna pojava redundancije. Rezultat obrade podataka u sloju brzih podataka jesu stvarnovremenski pogledi koji se inkrementalno ažuriraju pojavom novih podataka [19]. U [18] spomenut je Apache Spark kao moguća podloga za implementaciju sloja brzih podataka.

Slojevi serija podataka i brzih podataka, prema svojoj naravi, zajednički tvore cjelinu slojeva obrade i pohrane podataka.

Sloj posluživanja ima zadaću spajanja pogleda iz slojeva serija podataka i sloja brzih podataka, kako bi stvarali jedinstveni model pristupa obrađenim podacima [19]. Na taj način moguće je kreirati upite prema podacima neovisno o sloju u kojem se nalaze, kako bi se kompleksnost sustava i obrade podataka sakrila od korisnika.

Kappa arhitektura predstavlja pristup obradi podataka koji je znatno pojednostavljen i okrenut isključivo stvarnovremenskoj obradi. U navedenoj arhitekturi, za razliku od Lambda arhitekture, sloj serija podataka ne postoji. Shema Kappa arhitekture prikazan je na slici Slika 10:


Slika 10. Kappa arhitektura

Sličnost Kappa i Lambda arhitekture vidljiva je kroz postojanje sloja brzih podataka i sloja posluživanja koji zadržavaju istu namjenu. [18] navodi kako je gotovo nemoguće s performansne strane usporediti obje arhitekture, budući da njihovo korištenje ovisi o konačnoj primjeni. U [17] dana je usporedba promjene paradigme obrade serijskih podataka kao obrade toka povijesnih podataka, što je dovelo do motivacije za Kappa arhitekturu.

S tehničke strane, pored ranije navedene implementacije u [18] i [19] opširno je opisana implementacija pojedinih slojeva s fokusom na korištene tehnologije, pri čemu je za sloj serija podataka odabrana baza Cassandra, a za sloj brzih podataka Apache Spark. U [20] korištene su jednake tehnologije kao i u [19] za implementaciju sloja serija podataka, ali bez specificiranja implementacije sloja brzih podataka. Zanimljivost u pristupu iz [20] leži u implementaciji sustava odlučivanja o pokretanju obrade toka podataka ovisno o vremenskim parametrima izvođenja serijske obrade podataka. Autori u [21] implementaciju lambda arhitekture temelje na servisima u oblaku kao potporu obradi senzorskih podataka te su ujedno pokazali uštede korištenjem računarstva u oblaku. U radu [22] implementirana je lambda arhitektura u jezeru podataka kao potporu analizi pametnih elektroenergetskih mreža korištenjem Hadoop platforme u oblaku.

## 2.2. Istraživanja u domeni metapodataka u jezerima podataka

Rad [23] predstavlja arhitekturu skladišta podataka izvedenu nad velikim podacima koja se može prilagoditi korisničkim zahtjevima i promjenama u izvorima podataka. Model koji omogućava pohranu metapodataka kroz opis shema izvorišnih skupova podataka i njihovih evolvirajućih promjena nužan je za funkcioniranje predstavljene arhitekture.

Kao osnova za istraživanje uzeta je klasifikacija iz rada [24] koja nadopunjuje predloženi model metapodataka te predloženi formalni model metapodataka za jezera podataka [25]. Uz njih, model metapodataka nastavak je rada [26] koji predlaže tri vrste metapodataka (strukturne, svojstvene i semantičke) i predlaže način dohvata metapodataka iz heterogenih skupova podataka. Rad također koristi termin brzog podatkovnog puta (eng. *data highway*), koji označava cjevovod obrade podataka kakav se koristi i u skladištima podataka temeljenima na relacijskim bazama podataka [27].

Predložena arhitektura skladišta velikih podataka [23] kao jednu od ključnih komponenti identificira skladište metapodataka koje sadrži šest međusobno povezanih vrsta metapodataka. U svrhu praćenja evolucije velikih podataka, fokus istraživanja je na shematskim metapodacima, metapodacima preslikavanja i evolucijskim metapodacima. Shematski metapodaci opisuju sheme skupova podataka, metapodaci preslikavanja definiraju logiku ELT procesa, dok evolucijski metapodaci sadrže podatke o promjenama u izvorima podataka, a koji se mogu automatski dohvaćati tijekom ELT procesa ili iz specifikacije izvora podataka. U ostale vrste spadaju metapodaci o analitičkim kockama, metapodaci pravila promjena te metapodaci o potencijalnim promjenama sheme skladišta podataka.

Sam model opisan je kroz klase, podijeljene u tri grupe koje su bile u fokusu istraživanja.

Grupa klasa vezanih uz shemu gradi se oko klase *skup podataka*, koja je izgrađena od klase *podatkovne stavke*. Skup podataka sadrži potklase strukturiranih, polustrukturiranih i nestrukturiranih skupova podataka koji se, u pravilu, iskazuju kao jedna instanca klase *podatkovna stavka*. Dodatne informacije o takvom skupu podataka iskazane su kao dodatne stavke unutar skupa podataka. Skup podataka može se dobiti iz izvora podataka ili razine brzog podatkovnog puta te se ujedno opisuje informacijom o brzini dolaska novih podataka u njemu. Podatkovne stavke mogu se povezivati unutar istih skupova podataka ili između različitih skupova podataka (koji mogu biti i različitih potklasa) kroz refleksivnu klasu relacija. Metapodaci o porijeklu skupova podataka kroz brzi podatkovni put opisani su klasom

preslikavanja koja opisuje način transformacije podataka (sadržanog u atributu operacije) iz izvorišnog u odredišni skup podataka.

U grupi klasa koje opisuju svojstva nalaze se klasa *svojstva metapodataka* u kojoj su pohranjene različite karakteristike modela metapodataka u obliku para ključ-vrijednost. Vrijednosti karakteristika mogu se nalaziti u različitim domenama, stoga je važno da se mogu pohranjivati neovisno o njihovom sadržaju. Pojedine vrijednosti karakteristika mogu se dobiti automatskim dohvatom kroz analize skupova podataka, dok se ručno detektirane i unesene karakteristike povezuju s korisnikom koji ih je unio kroz klasu *autor*.

Grupa klasa povezanih uz evolucijske metapodatke sadrži informacije o evoluciji shema izvora podataka, skupova podataka i njihovih dodatnih značajki. Evolucija se iskazuje kroz klasu *promjena* koja se spaja s ostalim klasama, a svaka promjena u evoluciji zapisuje se uz pripadnu vremensku oznaku, vrstu promjene i enumerirani status evolucije. Potpora evoluciji jezera podataka također je definirana i podržanim skupom atomičnih promjena (eng. *atomic changes*), poput dodavanja izvora podataka, promjene oblika skupa podataka i sličnih. Kada se promjena u evolucijskim metapodacima detektira automatski, zapisuje se zajedno s pripadnim metapodacima u repozitorij kao instanca klase *promjena* sa statusom novih metapodataka.

U radu [8] predstavljen je model za upravljanje metapodacima u jezerima podataka temeljen na graf bazi podataka. Uz model, predložen je skup kriterija pomoću kojih je moguće evaluirati sustave za upravljanje metapodacima u jezerima podataka temeljenih na popisu očekivanih značajki sustava te tipologija metapodataka.

U uvodu rada predočen je koncept jezera podataka i povezanih istraživanja kao rješenja heterogenosti podataka, iznesen je problem izostanke potpore za nestrukturirane podatke i dana je definicija jezera podataka temeljena na dosadašnjim istraživanjima. Vezano uz evaluaciju sustava za upravljanje metapodacima, identificiran je problem pri uspostavljanju jasnih mjera efikasnosti samog sustava.

Predloženi model, nazvan MEDAL (Metadata model for Data Lakes) baziran je na hipergrafu u kojem je pojedini objekt reprezentiran kao hiperčvor u bazi podataka. Kao i prethodno navedeni rad [23], koristi klasifikaciju metapodataka kao interobjektnih i intraobjektnih metapodataka, a na temelju ranije predloženog u [30], te ih proširuje klasifikacijom globalnih metapodataka.

Intraobjektni metapodaci u predloženom modelu klasificiraju se na reprezentacije, koje su implementirane u jedan hiperčvor i sadrže atribute pojedinog skupa podataka, zatim transformacije, implementirane kao hiperbridove koje opisuju promjene pojedinih reprezentacija i sadrže atribute promjena te verzije koje prate promjene evoluciju jezera podataka kroz evoluciju samih objekata. Pojedini objekt, pohranjen u hiperčvoru, može sadržavati više reprezentacija dobivenih iz drugih objekata putem transformacija, što ujedno nudi i mogućnost praćenja porijekla podataka. Kombiniranjem transformacija i verzija podataka prati se povijest promjena pojedinog objekta u jezeru podataka, te se unutar hiperčvora generira stablo povijesti čiji je korijen inicijalna reprezentacija.

Interobjektni metapodaci iskazuju se hiperbridovima između dva ili više čvora, gdje svaki brid korespondira skupu objekata (čvorova), a grupiranje čvorova ovisi o broju disjunktnih vrijednosti u pojedinom grupirajućem atributu. Sličnost između objekata iskazuje se neusmjerenim težinskim bridom s atributima koji označavaju pojedine značajke sličnosti. Svaki hiperbrid može biti derivat njegovog roditeljskog hiperbrida, pri čemu se koriste usmjereni hiperbridovi koji naznačavaju hijerarhijsku strukturu nasljedstva te su opisani atributima.

Globalni metapodaci obuhvaćaju specifične elemente koji ne spadaju u interobjekte ni u intraobjektne metapodatke, no mogu se po potrebi koristiti. Primjeri globalnih metapodataka su dnevnički zapisi i indeksi nad pojedinim atributima, usko vezani uz tehničku stranu implementacije skladišta podataka.

U svrhu usporedbe s ostalim sustavima za upravljanje metapodacima, predložen je skup značajki pojedinog sustava na temelju kojih se može kvalitativno ocijeniti mogućnost sustava. Predložene značajke su semantičko obogaćivanje, indeksiranje podataka, kreiranje i konzervacija veza, podatkovni polimorfizam, verzioniranje podataka i praćenje korištenja. Zapaženo je da se posljednje dvije značajke često spajaju u jednu značajku – praćenje porijekla – no kako imaju različite uloge u sustavima za upravljanje metapodacima, predlaže se da se evaluiraju odvojeno.

Temeljem navedenih značajki, autori su proveli kvalitativnu analizu poznatih dostupnih sustava i podijelili ih pritom na konceptualne sustave (modele metapodataka) i sustave koji imaju mogućnost operativne implementacije u jezerima podataka. Cilj analize je, pored istraživanja postojećih sustava, bio pokazati kompletnost predloženog MEDAL sustava na

način da pokriva svih šest predloženih traženih značajki sustava, dok ostali poznati sustavi obuhvaćaju pet ili manje značajki.

U radu [28] predstavljen je pristup upravljanju metapodacima u jezerima podataka baziran na generičkoj i proširivoj klasifikaciji metapodataka. Kako se radi o sustavu koji je potpora jezerima podataka, njegova implementacija uključuje rad sa strukturiranim, polustrukturiranim i nestrukturiranim tipovima podataka.

U istraživanju povezanih radova, autori su se fokusirali na klasifikacije metapodataka u jezerima podataka, uzevši u obzir dosadašnje poznate klasifikacije metapodataka u skladištima podataka. Podjela na kojoj se i temelji predloženi model, jest na intra-metapodatke, koji opisuju skup podataka, te inter-metapodatke koji opisuju odnose između različitih skupova metapodataka. Alternativno, podjela metapodataka može biti i na tehničke (vrste podataka, oblik podataka i struktura), operativne (podaci o obradama) te poslovne (poslovni objekti i njihova značenja).

Predloženi model integrira inter-metapodatke i intra-metapodatke te se na temelju prethodnog rada [24] predlaže dodatna klasifikacija integriranih metapodataka:

Intra-metapodaci, temeljeni na prethodnim istraživanjima, dijele se u metapodatke obilježja skupova podataka, definicijske metapodatke – koji sadrže još jednu razinu podjele na semantičke i shematske metapodatke, navigacijske metapodatke s podacima o lokacijama skupova podataka, metapodatke o podrijetlu (eng. *lineage*) iz kojih iščitavamo izvorište skupa podataka i obilježja obrada podataka. Ujedno je model metapodataka nadograđen metapodacima o pristupu koji sadrže

- podatke o alatima pomoću kojih se pristupa skupovima podataka
- korisnicima tih alata za pristup skupovima podataka
- metapodatke o kvaliteti skupova podataka te
- sigurnosne metapodatke.

Inter-metapodaci se prema prethodnim istraživanjima klasificiraju u metapodatke o zadržavanju (eng. *containment*) u kojima su označeni podskupovi i nadskupovi podataka, metapodatke o međusobnim izvorima (eng. *provenance*) koji naznačavaju porijeklo jednog skupa podataka iz drugog skupa podataka, metapodatke o logičkim grozdovima (eng. *cluster*) skupova podataka na temelju kojih se definiraju domenski skupovi podataka i metapodatke o

sličnosti sadržaja kroz koje se održavaju podaci o skupovima podataka koji dijele iste atribute. Predloženi model dodatno sadrži i klasu metapodataka o djelomično preklapajućim atributima u različitim skupovima podataka, a u kojima su sadržani odgovarajući podaci.

Primjena predloženog modela metapodataka temelji se na podjeli jezera podataka na četiri zone: zonu sirovih podataka (eng. *raw data zone*) koja sadrži podatke u njihovom izvornom obliku, zonu obrade podataka (eng. *process zone*) u kojoj se sirovi podaci obrađuju i pohranjuju u međuspremnik, zonu pristupa podacima (eng. *access zone*) u kojoj se nalaze dostupni obrađeni podaci te zona upravljanja podacima. Autori navode da predloženi model sadrži niz prednosti: od potpore za strukturirane i nestrukturirane podatke, pohrane svih vanjskih skupova podataka, zapisa svih akcija u jezeru podataka, podacima o svakom pojedinom skupu podataka do kontrole nad kvalitetom, osjetljivosti i razinama pristupa podacima.

Konačno, dan je pregled implementacije predloženog modela u postojećem jezeru podataka te se navode dvije različite implementacije temeljene na relacijskoj bazi podataka te graf bazi podatak, uz navođenje prednosti oba pristupa implementaciji.

Uzevši u obzir razvoj interneta stvari, uz sve moćnije procesore ugrađene u pomične uređaje, razvila se nova disciplina računarstva na rubu (eng. *edge computing*) koja omogućuje obradu izvorišnih podataka u određenoj mjeri na samom uređaju. Na tom tragu, razvijen je okvir za rad s metapodacima na rubu u okviru laguna podataka (eng. *Data Lagoon*), pristupu integraciji jezera podataka s komponentama računarstva na rubu [29].

Koncept lagune podataka je primjena arhitektura jezera podataka na čvorove (uređaje) u mreži, pri čemu svaki uređaj ima svoj repozitorij heterogenih podataka koje može lokalno obraditi. Budući da je repozitorij metapodataka integralni dio arhitekture jezera podataka, što uključuje i lagune podataka, predložen je programski okvir za upravljanje metapodacima na spoju jezera podataka i laguna podataka i dana je prijedlog njegove arhitekture na visokom nivou (eng. *high level*). Predložena arhitektura repozitorija metapodataka orijentirana je na slučajeve kad se lagune podataka integriraju s udaljenim jezerima podataka koja se nalaze na višoj razini u podatkovnom cjevovodu.

Arhitektura se, pored repozitorija podataka na obje strane, sastoji od dva ključna dijela: uslužne osnove (eng. *service plane*) i podatkovne osnove (eng. *data plane*). Uslužna osnova uključuje sve procese upravljanja koji omogućuju povezivanje laguna i jezera podataka na način da su jezera podataka izvor za dio podataka koje lagune konzumiraju. U tu svrhu predlaže se

komponenta brokera s osnovnom zadaćom upravljanja komunikacijama između obje strane uslužne osnove , a ujedno u repozitoriju metapodataka održava registar svih dostupnih servisa povezanih kroz uslužnu osnovu.

Podatkovna osnova ima namjenu kompletiranja kontinuiteta toka podataka i informacija između izvora podataka i repozitorija velikih podataka. Lagune podataka koriste podatkovne osnove kako bi omogućile neprekidnu integraciju kroz više kanala tokova podataka različitih osobina, što je analogno lambda arhitekturi toka podataka u sustavima velikih podataka. Na taj način ostvaruje se mogućnost prijenosa obrađenih podataka od laguna prema jezeru podataka kroz komponentu rubne obrade podataka, ali istovremeno pruža i mogućnost dohvata izvornih, nepromijenjenih podataka s laguna.

Skup predloženih dimenzija u modelu metapodataka za lagune podataka sastoji se od infrastrukturnih resursa, skupova podataka, informacija, skupa primljenjivih operacija nad podacima, sigurnosnih metapodataka, modela potrošnje i stanja izvođenja. Posebno je zanimljiva dimenzija skupa primjenjivih operacija, gdje se pored standardnih operatora integracije podataka specificiraju logičke operacije poput agregacija tokova podataka nad prozorom, pročišćivanja skupa podataka, obogaćivanja skupova podataka i profiliranja skupova podataka.

## 2.3. Istraživanja u domeni polustrukturiranih podataka

Polustrukturirani podaci mogu se iskazati kao skup podataka u slaboj shemi (eng. *weak schema*). Kroz slabu shemu moguće je iskazivanje strukture podataka s varijabilnom shemom, no s mogućnosti kreiranja konačne sheme pomoću odgovarajućih operacija.

U [45] predložen je model za definiciju polustrukturiranih podataka korištenjem liste parova oblika atribut-vrijednost (eng. *Attribute-Value Pairs List*), pri čemu se svaki par sastoji od dvije ntorke (atribut i vrijednost). U nastavku ovog rada, lista parova oblika atribut vrijednost označava se s LPAV. Definicije u nastavku ovog poglavlja koje se odnose na polustrukturirane podatke preuzete su iz [45].

Pojedinačni par atribut-vrijednost tvori listu parova atribut-vrijednost

$$(a \in A) \wedge (v \in V) \rightarrow \{(a, v)\} \in D$$

pri čemu je **skup lista parova atribut-vrijednost (LPAV)** označen s $D$, skup atributa s $A$ i skup vrijednosti s $V$.

Unija dvaju skupova LPAV ($D_1$ i $D_2$) također tvori skup LPAV

$$D_1, D_2 \in D \rightarrow D_1 \cup D_2 \in D$$

Svaki atribut predstavlja uređeni skup jedne ili više varijabli, pri čemu je svaka pojedina varijabla ujedno atribut.

1) Kada je S skup nizova znakova (*eng. string*), atribut se definira na sljedeći način: pojedinačni *(eng. singleton)* atribut $s \in S \rightarrow s \in A$
2) Složeni atribut (atribut s više varijabli) $a_1, a_2, \ldots a_n \in S \rightarrow (a_1, a_2, \ldots a_n) \in A$, pri čemu je $(a_1, a_2, \ldots a_n)$ uređeni niz varijabli atributa.

Vrijednost je instanca, ili skup instanci, koja se pridružuje korespondentnom atributu i njegovim varijablama.

Pridruživanje vrijednosti atributima definirano je na sljedeći način:

1) Pojedinačni atribut i vrijednost: $a \leftarrow v$, pri čemu je $a \in A$ i $v \in V$
2) Složeni atribut i vrijednost:$(a_1, a_2, \ldots, a_n) \leftarrow (v_1, v_2, \ldots, v_n)$, pri čemu je $(a_1, a_2, \ldots a_n) \in A$, $(v_1, v_2, \ldots, v_n) \in V$, a svaki $v_i$ je pridružen $a_i$, $(1 \leq i \leq n)$

Domene atributa sadrže jednostavne nizove, reference na vrijednosti, skupove ili liste vrijednosti te objekte tipa LPAV. Zbog toga objekt tipa LPAV, ili skupa LPAV objekata, može biti komponenta nekog drugog LPAV objekta, čime je omogućena ugniježđena struktura

Za skup vrijednosti $V$ definirani su tipovi vrijednosti:

1) Jednostavan niz znakova $s \in S \rightarrow s \in V$, pri čemu je $S$ skup nizova znakova.
2) Referenca na bilo koji tip vrijednosti $v \in V \rightarrow \&v \in V$, pri čemu je $\&v$ referenca na $v$, odnosno identifikator od v
3) Skup bilo kojih tipova vrijednosti $v_1, v_2, \dots, v_n \in V \rightarrow \{v_1, v_2, \dots, v_n\} \in V$
4) Lista bilo kojih tipova vrijednosti $v_1, v_2, \dots, v_n \in V \rightarrow < v_1, v_2, \dots, v_n > \in V$
5) LPAV objekt $d \in D \rightarrow d \in V$, pri čemu je $D$ skup LPAV objekata
6) Null (prazna vrijednost)
7) Identifikator - "a self contained label", niz znakova koji započinje s '#', kojeg drugi LPAV objekti koriste kao referencu.

Atomarne vrijednosti u skupovima LPAV isključivo su nizovi znakova. Ostali tipovi podataka iz sustava tipova podataka prema [45] nisu podržani. Model predložen u [45] predviđa pretvaranje u druge atomarne tipove podataka, što uvodi dodatni sloj kompleksnosti. Zadržavanje atomarnih vrijednosti u nizu znakova može se primijeniti na sloj pohrane podataka ukoliko se radi o ograničenju odabrane baze podataka.

### 2.3.1. Operacije konstrukcije shema

Shema je definirana pomoću klasa i njihovih međusobnih odnosa. Odnosi između klasa ujedno impliciraju odnose između objekata. Objektna shema podrazumijeva shemu unutar objektno orijentirane pohrane podataka.

Moguće su dvije vrste odnosa između objekata:

- *jest (eng. is-a)*, koja je temelj kompozicijske hijerarhije i

- *jest dio (eng. is-part-of)*, koja je temelj hijerarhije nasljeđivanja

Na temelju objekata definira se tip, koji se odnosi na kolekciju objekata istih strukturalnih i ponašajnih informacija u objektno-orijentiranom modelu. Tipovi se implementiraju kao klase, dok klasa definira odnos kolekcija.

Shema se stvara zasebnim skupom operatora iz jednog ili više skupova LPAV. Skup operatora stvaranja shema sastoji se od operatora stvaranja klasa iz kolekcije instanci, operatora

spajanja, operatora kompozicije, operatora inkluzije i operatora dobivenih trivijalnom inverzijom definiranih operatora.

1) Operator stvaranja klase korištenjem kolekcije instanci

Za objekte skupova LPAV $S_1, \dots, S_m$ i njihove pripadne skupove atributa $A_1, \dots, A_m$, klasa objekata skupa LPAV $\mathcal{U}$ definira se kao $\mathcal{U} = \{S_1, S_2, \dots, S_m\}$ , pri čemu je skup atributa $\mathcal{U}_A = A_1 \cup A_2 \cup \dots \cup A_m$. Ako postoji neki LPAV objekt p kod kojeg a nije iz $A_i$, vrijednost atributa $a$ postavlja se na praznu vrijednost (*null*).

Rezultat operatora *Object_Collect* $(S_1, S_2, \dots, S_m)$ je klasa objekata skupa LPAV $\mathcal{U}$.

U operatorima spajanja temeljenih na skupovima LPAV razlikujemo operatore spajanja objekata i operatore spajanja klasa.

2) Operator spajanja objekata

Za dva objekta skupa LPAV $S$ i $T$, novi skup LPAV $U$ rezultat je operatora *Object_Merge* $(S, T)$ ako vrijedi

$$U = \{w | (w \in S \cup T) \wedge \exists\, \hat{a} (\hat{a} \in S \wedge \hat{a} \in T)\}$$

pri čemu je skup atributa skupa LPAV $U$ jednak $S \cup T$, a $\hat{a}$ zajednički (dijeljeni) par atribut-vrijednost skupova LPAV $S$ i $T$. $w$ označava par atribut-vrijednost.

3) Operator spajanja klasa

Za dvije klase skupa LPAV $\mathcal{S}$ i $\mathcal{T}$, nova klasa skupa LPAV $\mathcal{U}$ rezultat je operatora *Class_Merge* $(\mathcal{S}, \mathcal{T})$ ako je $U \in$ rezultat operatora *Object_Merge* $(S, T)$ i ako je $U \in \mathcal{U}$ rezultat operatora *Object_Collect* $(U_1, U_2, \dots, U_m)$, pri čemu je $S \in \mathcal{S}$, $T \in \mathcal{T}$ i $U_i \in \mathcal{U}$ za $1 \leq i \leq m$

Operatori kompozicije rezultiraju novim skupom LPAV, pri čemu dva skupa LPAV, argumenti operatora, u presjeku imaju minimalno jedan par atribut-vrijednost. Kompozicija označava odnose *jest dio*.

4) Operator kompozicije objekata

Za dva skupa LPAV $S$ i $T$, novi skup LPAV $U$ rezultat je operatora *Object_Compose* $(S, T)$ ako vrijedi

$$U = (S \setminus t) \cup \hat{T}$$

pri čemu je $t \in S$ i $t \in T$, a $\hat{T}$ se odnosi na sam skup LPAV $T$ ili je referenca na skup LPAV $T$. Skup atributa od $U$ jednak je skupu atributa od $S \cup T$.

5) Operator kompozicije klasa

Za dvije klase skupa LPAV $\mathcal{S}$ i $\mathcal{T}$, nova klasa skupa LPAV $\mathcal{U}$ rezultat je operatora *Class_Compose* $(\mathcal{S}, \mathcal{T})$ ako je $U \in$ rezultat operatora *Object_Compose* $(S, T)$ i ako je $U \in \mathcal{U}$ rezultat operatora *Object_Collect* $(U_1, U_2, \dots, U_m)$, pri čemu je $S \in \mathcal{S}$, $T \in \mathcal{T}$ i $U_i \in \mathcal{U}$ za $1 \leq i \leq m$

6) Operator inkluzije

Operator inkluzije odnosi se na odnose *jest*.

Za dva skupa LPAV $\mathcal{U}$ i $\mathcal{V}$, novi odnos $\mathcal{U}$ *je podskup od* $\mathcal{V}$, dobiva se kao rezultat operatora *Class_Include* $(\mathcal{U}, \mathcal{V})$ ako vrijedi $\mathcal{U} \subseteq \mathcal{V}$, pri čemu za skupove atributa $\mathcal{U}_{\mathcal{A}}$ i $\mathcal{V}_{\mathcal{A}}$, koji pripadaju skupovima LPAV $\mathcal{U}$ i $\mathcal{V}$, vrijedi $\mathcal{U}_{\mathcal{A}} \subseteq \mathcal{V}_{\mathcal{A}}$.

## 2.4. Istraživanja u domeni prostorno-vremenskih tipova podataka i operacija

U [33] predloženi su tipovi podataka i operacija za rad s prostorno-vremenskim tokovima podataka. Inicijalno se razmatraju modeli podataka i pripadni upitni jezici, podijeljeni u tri grupe: bazirani na ograničenjima, kompozitni prostorno-vremenski tipovi podataka te bazirani na ortogonalnosti prostora i vremena. Interoperabilnost prostornih, vremenskih i prostorno-vremenskih baza podataka omogućena je pomoću zajedničkog sloja upitnih jezika.

U jednostavne tipove podataka ubrajamo osnovne tipove podataka (base - *int, real, string, bool*), prostorne tipove podataka (spatial - *point, line, region*) i vremensku oznaku (time - *instant*). Konstruktori nad jednostavnim tipovima podatka tvore nove ciljane vrste: *range* - kombinaciju osnovnih tipova i vremenske oznake (konstruktor *range*) i *temporal* – kombinaciju osnovnih i prostornih tipova podataka (konstruktori *moving, intime*). Kao osnova objektno-relacijske algebre koriste se ciljane vrste identifikatora (*identifier* – konstruktor *id)*, ntorke (*tuple* – konstruktor tuple) i odnosa (*relation* – konstruktor *relation*). Kako bi se tokovi podataka modelirali analogno objektno-relacijskim modelima, koriste se ciljane vrste *stuple* s konstruktorom *stuple* (koja sadrži podatke o lokaciji pokretnog objekta u određenom vremenskom trenutku) te *stream* s konstruktorom *stream*, temeljenom na ciljanoj vrsti *stuple*.

U sustavu za upravljanje tokovima podataka definirane su tri grupe operatora: *stream-to-relation (*operatori na klizećem prozoru temeljeni na SQL-99 standardu*), relation-to-stream* (temeljeni na standardnim relacijskim operatorima) i *relation-to-relation*. Svi standardni relacijski operatori dio su sustava za upravljanje bazama pokretnih objekata uz operatore nad tipovima podataka. Operacije se prvo definiraju nad jednostavnim tipovima podataka (osnovni i prostorni tipovi), a zatim se proširuju na pripadne kompleksne tipove podataka (vremenski).

U svrhu obrade neograničenih tokova podataka koriste se operatori klizećih prozora. Granice klizećih prozora mogu biti vremenske (vremenski okvir) ili temeljene na određenom broju ntorki. Operatori klizećih prozora kao ulaz prihvaćaju tok podataka, dok je izlaz relacijski. Od operatora klizećih prozora s vremenskim granicama definirani su NOW i PAST, dok su operatori temeljeni na ntorkama UNBOUNDED, SINCE i RANGE.

U radu [34] formaliziran je programski okvir za rad s geoprostornim tokovima podataka kroz specifikaciju tipova podataka i operacija nad njima i dan primjer implementacije na temelju korisnički definiranih agregatnih funkcija. Specificirane su značajne razlike između sustava za upravljanjem tokovima podataka i tradicionalnih sustava za upravljanje bazama podataka:

- Tokovi podataka su nizovi zapisa, poredani po vremenu pristizanja ili drugom ključu poretka (primjerice, vremenu stvaranja zapisa) koji postaju dostupni za obradu tokom vremena, za razliku od baza podataka gdje su dostupni prije obrade
- Tokovi podataka nemaju kontrolu nad poretkom pristizanja podataka
- Podaci se stvaraju neprekidno, što znači da su tokovi podataka neograničeni ili imaju nepoznatu duljinu

Predloženi sustav tipova podataka temelji se na povijesnim i trenutnim upitima, korištenjem višestruko razvrstane algebre (*many-sorted algebra*) i signatura drugog reda (second-order signature). Na apstraktnoj razini promatramo pomičnu točku (*moving point*) u tri (Euklidski prostor $E^2$ u vremenu) odnosno četiri dimenzije (Euklidski prostor $E^3$ u vremenu), odnosno preslikavanje iz neprekinute vremenske domene u neprekinutu prostornu domenu.

U višestruko razvrstanoj algebri razlikujemo skupove i funkcije. Oznaka je par skupova (*S,* Ω) koji se nazivaju vrste (S) i operacije (Ω).

Pojedina operacija (označena s *n*) provodi se nad (k+2) ntorki:

$n : s_1 \; x \; \ldots \; x \; s_k \; \rightarrow s$, gdje su $s_1, \ldots, s_k, s \in S$ vrste i $k \geq 0$.

Za slučaj kad je k = 0, operacija se naziva *konstanta vrste s*.

Oznaka drugog reda sastoji se od dvije uparene višestruko razvrstane oznake, pri čemu prva oznaka definira sustav tipova podataka, a druga oznaka definira zbirku polimorfnih operacija nad tipovima podataka iz sustava tipova podataka.

U svrhu potpore praćenju geoprostornih podataka tipa γ koji se mijenjaju kroz vrijeme, predložena je vremenska funkcija koja je element tipa

t (γ) = vrijeme → γ

Vremenske funkcije su osnova algebarskog modela prostorno-vremenskih tipova podataka, pri čemu je γ jedan od geoprostornih tipova podataka *point, multiPoint, line* ili *polygon*, a rezultat funkcije su novi tipovi podataka *movingPoint* (τ (*point*)), *movingMultiPoint* (τ (*multiPoint*)), *movingLine* (τ (*line*)) i *movingPolygon* (τ (*polygon*)).

Analogno tome, prošireni su i osnovni tipovi podataka u pokretne osnovne tipove: *movingInteger, movingBoolean, movingFloat* i *movingInstant* koji imaju svrhu u prostorno-vremenskom kontekstu.

U radu [33] definirane su apstraktne semantike vrsta BASE, SPATIAL, TIME, RANGE i TEMPORAL. Primjenom vremenski ovisnog konstruktora (RANGE, TEMPORAL) konstruktora na jednostavne tipove podataka dobivaju se novi tipovi podataka. Dodatne specifičnosti tipova podataka su:

- Tip podataka *instant*, koji se odnosi na vremenske oznake specificiran je kao izomorfan s realnim brojevima.
- Konstruktor *range* može se primijeniti na tipove podataka BASE i TIME, sadržava početnu i krajnju vrijednost i sadrži sve vrijednosti koje se nalaze između njih.
- Konstruktor nad tipovima podataka oblika TEMPORAL može se primijeniti na tipove podataka BASE i SPATIAL.

Vrste tipova podataka IDENTIFIER, TUPLE i RELATION, odnosno konstruktori *id, tuple* i *relation* su osnova objektno-relacijske algebre i kao takvi su osnova za izgradnju sustava za upravljanje prostorno-vremenskim bazama podataka.

**Apstraktna semantika tipova podataka u geoprostornim tokovima podataka**

Geoprostorni tokovi podatka razlikuju se od konvencionalnih tokova podataka baziranih na relacijskom modelu kroz dvije specifične značajke:

- Vrijeme valjanosti odnosno kreiranja ntorke u toku podataka definirano je vremenskim atributom $A\tau$
- Oblik i lokacija praćenih objekta ili objekata opisanih ntorkom u toku podataka definirani su geoprostornim atributom $A\gamma$.

U radu [37] analizirane su performanse pohrane heterogenih geoprostornih podataka u sustavima za pohranu prostornih podataka. Uspoređeni su sustavi pohrane strukturiranih podataka (ntorki) i polustrukturiranih podataka u JSON obliku zapisa. Usporedba je obavljena nad sustavima MongoDB i PostgreSQL, odnosno njegovog proširenja PostGIS. Rezultat analize pokazao je kako nema značajnih performansnih degradacija u radu s velikim količinama podataka korištenjem postrelacijskih baza podataka. Kao prethodnik tome istraživanju značajan je i rad [38] koji je usporedio postrelacijske, dokument baze podataka i graf baze podataka u

svrhu pohrane prostornih podatka i podržanih operacija. Izbor modela pohrane podataka pritom ujedno ukazuje na vrste baza podataka koje pružaju mogućnost rada s prostornim, odnosno prostorno-vremenskim tipovima podataka.

# 3. Model prilagođenog jezera podataka za obradu i pohranu prostorno-vremenskih tokova podataka

U poglavlju 1.1 dana je definicija jezera podataka kao repozitorija heterogenih podataka pohranjenih u nepromijenjenom obliku. Striktno uzevši u obzir, takva definicija odgovara dvozonskoj arhitekturi jezera podataka koja je bila inicijalno definirana kao referentna arhitektura. Slijedom takve arhitekture, u svrhu analize podataka nije bio mogući drugačiji pristup osim sheme na upit (eng. *schema-on-read*), bez mogućnosti standardiziranih analiza koje bi zadržale jednaku strukturu zapisa i rezultata.

Dvozonska arhitektura dostatna je za korištenje jezera podataka, no ostavlja mnogo neriješenih situacija – od specifikacije obrade podataka nakon njihovog prihvata, definicije opsega pristupa podacima, preko procesa upravljanja podacima unutar jezera podataka do podrške za poslovne aplikacije.

Uvođenje višezonske arhitekture poput [42] i [43] rješava tek dio navedenih situacija, kao što su definiranje prava pristupa podacima, sljedivost promjena strukture podataka kroz zone i u određenoj mjeri procese upravljanja podacima. Istovremeno, izostaje eksplicitno definiranje repozitorija metapodataka i podatkovnih cjevovoda pomoću kojih se obavljaju procesi gutanja podataka.

U slučaju korištenja višeslojne arhitekture [41], definirani su procesi gutanja podataka, ekstrakcije i pohrane metapodataka, pohrane i promjene podataka te sloj koji sakriva preostale slojeve od korisnika.

Kombinacija obje arhitekture daje konačnu sliku potreba definicije jezera podataka: načine obrade i pohrane podataka, načine pristupa podacima i pohranu metapodataka. Specifičnosti ekstrakcije podataka nije potrebno eksplicitno definirati, budući da ti procesi ne definiraju samo jezero podataka.

Pored definiranja arhitekture jezera podataka, potrebno je prilagoditi postojeće poznate sustave tipova podataka. Jedna od odlika jezera podataka je potpora za rad s nestrukturiranim podacima, koji u postojećim sustavima tipova podataka nisu bili eksplicitno definirani. Jedan od doprinosa ovog rada jest i specificiranje sustava tipova podataka u svrhu prilagodbe rada s jezerima podataka.

Jezera podataka u velikoj mjeri oslanjaju se na rad s nestrukturiranim podacima koji su svojim volumenom značajno veći od strukturiranih i polustrukturiranih. Iz tog razloga,

postojeći sustavi tipova podataka trebaju se proširiti u svrhu potpore radu s nestrukturiranim tipovima podatka. Nestrukturirani podaci, uz izostanak metapodataka, pripadaju tipu podataka *blob*. Tek kroz dodatne obrade (pridodavanje metapodataka, prilagodba strukture) nestrukturirani podaci postaju korisni kao izvori informacija i podloga za analize. Uz definiciju potpore tipu podataka *blob*, potrebno je proširiti i skup postojećih operacija.

Unutar jezera podataka, nakon pohrane nepromijenjenih podataka u perzistentnu memoriju, dio definiranih zona unutar arhitekture zahtijeva pohranu podataka u prilagođenom obliku. U navedenim zonama moguće je korištenje strukturiranih i polustrukturiranih zapisa – za odabir obje vrste zapisa postoje prednosti i nedostaci. Korištenje strukturiranih podataka je dobro poznato – svaka promjena strukture praćena je odgovarajućim naredbama i može se automatski opisati u repozitoriju metapodataka. Upitni jezici za rad sa strukturiranim podacima široko su rasprostranjeni i korišteni, a analitičke aplikacije dolaze s podrškom za rad s njima. Negativna strana korištenja strukturiranih podataka je, pak, nužnost integracije podataka. U svrhu potpore gotovo stvarnovremenskoj obradi podataka, nužne su značajne prilagodbe procesa integracije u odnosu na uobičajene obrade serija podataka.

S druge strane, korištenje polustrukturiranih zapisa omogućuje fleksibilno definiranje sheme podataka po potrebi. Podaci u okruženju velikih podataka na sučelja dolaze velikom brzinom, a struktura izvornih podataka, a samim time i analitički prilagođenih zapisa, brže se mijenja nego kod skladišta podataka. Uzevši takve značajke u obzir, polustrukturirani zapisi definitivno su bolji odabir za prilagođene zapise. Fleksibilna shema ujedno predstavlja i najveći rizik, odnosno nedostatak – uklanjanje atributa iz sheme može utjecati na analitičke procese. Rješenje takve situacije leži u uspostavljanju procesa upravljanja podacima, ponajprije kroz naglašeno praćenje promjena sheme u repozitoriju metapodataka. Pritom je važno da u zonama koje pružaju podršku analitičkim procesima u nijednom trenutku atributi ne smiju biti izostavljeni iz sheme, već se shema može isključivo proširivati. Evolucijskim praćenjem takvih promjena u repozitoriju metapodataka ujedno se i ostvaruje praćenje porijekla podataka unutar cijelog jezera podataka.

Uloga repozitorija metapodataka i njegova arhitektura važna je, pored procesa upravljanja podacima, i za definiranje podatkovnih cjevovoda, odnosno potpore stvaranju zona podataka. Prvenstveno, izvorni podaci moraju se dobro opisati metapodacima kako bi postojala referentna točka o strukturi nepromijenjenih podataka. Metapodaci o strukturi podataka iz izvora čine osnovu za praćenje porijekla podataka unutar jezera podataka, što je nužno za potporu stvaranju

ispravnih podatkovnih cjevovoda. Repozitorij metapodataka, pored strukture podataka, sadrži i podatke koji su potpora automatizaciji procesa gutanja, odnosno integracije podataka, ali i ostalim procesima upravljanja podacima.

## 3.1. Model – tipovi i operacije

U radu [31] postavljen je formalni okvir, zasnovan na konceptu algebri više sorti (eng. *many sorted algebra*). Njegovim korištenjem omogućeno je opisivanje podatkovnih modela i upitnih jezika, implementacijskih modela i izvršnih jezika i njima pridruženih optimizacijskih pravila.

Navedeni formalni okvir koristi koncept signature za specifikaciju apstraktnih tipova podataka. Pojedina signatura sastoji se od dvije skupine simbola, *sorta* i *operatora*, gdje je svaki operator označen nizom sorti. Formalni okvir definira pojam signature drugog reda (eng. *second-order signature*) koja predstavlja dvije povezane signature više sorti. U njoj prva signatura definira sustav tipova podataka, operatore koji su njihovi konstruktori i izraze koji predstavljaju tipove podataka. U drugoj signaturi koriste se izrazi prve signature kao sorte, dok se nad njima definiraju operatori.

Sustave tipova podataka može se opisati signaturama, pri čemu sorte nazivamo vrstama (eng. *kind*). Tipovi podataka u takvim signaturama predstavljeni su kroz izraze, dok operatori predstavljaju konstruktore tipova podataka. Nad definiranim sustavima tipova podataka moguće je definirati operacije. Dodatno, unutar sustava moguće je grupirati vrste u nove skupove tipova podataka, što otvara mogućnosti korištenja polimorfnih operacija te operatora nad tipovima podataka.

Osnova sustava apstraktnih tipova podataka i operacija temelji se na prethodnim istraživanjima [33] [34] [35] [36] korištenjem formalnog okvira iz [31]. Uz poznate tipove podataka podržane u sustavima za pohrane, u fokusu predloženog modela nalaze se prostorni tipovi podataka. Cilj je definirati skup tipova podataka koji odgovaraju OGC standardima uz mogućnost njihovog preslikavanja u vremensku domenu.

### 3.1.1. Sustav tipova podataka

Temelj formalnog okvira korištenog u modelu jezera podataka je signatura drugog reda, detaljnije opisana u radu [32]. Signaturu drugog reda definiramo kao uređenu petorku s elementima $(K, \Gamma, T, \Delta, \Omega)$:

- $K$ je skup koji se sastoji od vrsta (skupova tipova podataka)
- $\Gamma$ je $(K \cup T, K)$ - sortna proširena signatura. Operatori u Γ su konstruktori tipova, koji kao argumente prihvaćaju vrste i tipove podataka, a rezultiraju novom vrstom
- $T$ je skup koji se sastoji od tipova. Pojedini tip generira se iz elemenata vrste $K$, temeljem konstruktora tipova iz Γ.

- $\Delta$ je$(K \cup T, K)$ - sortna proširena signatura čiji su elementi operatori tipova
- $\Omega$ je $T$ – sortna signatura čiji su elementi operatori.

Na temelju signature drugog reda definiraju se sustav tipova podataka i operatori nad njima. Sustav tipova podataka definiran je elementima $K, \Gamma$ i $T$, dok su operatori definirani elementima $\Delta$ i $\Omega$.

Osnova prostornih tipova podataka je točka, dana u dvodimenzionalnoj ili trodimenzionalnoj domeni prostora. Dodavanjem domene vremena, točku iz euklidskog prostora podižemo u prostorno-vremensko okruženje, čime postaje pokretna točka. Pokretna točka se reprezentira neprekinutom krivuljom u trodimenzionalnoj ili četverodimenzionalnoj prostorno-vremenskoj domeni. Prostorno-vremenske podatke, pak, promatramo kao preslikavanje skupa točaka iz vremenske domene u prostornu domenu.

Korištenjem navedenog formalnog okvira stvoren je temelj sustava tipova podataka i operacija nad njima, opisan u radovima [32] [33] [34]. U svrhu potpore prostorno-vremenskih tokova podataka u jezerima podataka predloženi sustav je proširen kako je opisano u poglavlju 3.1.2.

### 3.1.2. Proširenje sustava tipova podataka

Kao potporu pohrani prostorno-vremenskih tokova podataka potrebno je predvidjeti i tipove podataka koji omogućuju pohranu nestrukturiranih podataka. Nestrukturirane podatke pritom trebamo promatrati u različitim kontekstima, pridjeljivanjem njima najbliže strukture. Primjerice, nestrukturirani zapis koji se sastoji isključivo od riječi i eventualnih numeričkih vrijednosti potrebno je prethodno identificirati kao tekst. Analogno, nestrukturirani zapisi poput slika ili videozapisa zapisuju se u izvornom obliku koji odgovara tipu *blob* (eng. *binary large object*). Korištenjem tipa *blob* moguće je pohraniti bilo koji zapis, ponajviše one čija se struktura ne može opisati ostalim poznatim tipovima podataka. Dodatno, veliki broj sustava za obradu podataka podržava rad s tipom podataka *blob*. Kod sustava bez ugrađene potpore za rad s *blob* tipom podataka, podatak se može pridružiti zapisu putem hiperveze na datoteku unutar datotečnog sustava.

Sustav tipova podataka za potporu prostorno-vremenskim tokovima podataka prikazan je u tablici Tablica 1.

Tablica 1. Sustav tipova podataka

| Konstruktor tipa | Signatura | Sorta (vrsta) |
|---|---|---|
| int, real, string, bool, blob | | BASE |
| point, multiPoint, line, polygon, region | | SPATIAL |
| instant | | TIME |
| range | $BASE \cup TIME$ | RANGE |
| moving, intime | $BASE \cup SPATIAL$ | TEMPORAL |
| id | | IDENTIFIER |
| tuple | $\left(id \times \left((BASE \cup SPATIAL \cup TIME \cup RANGE \cup TEMPORAL\right)\right)^{+}$ | TUPLE |
| relation | *TUPLE* | RELATION |
| stuple | $\left(\left(id \times \left((BASE \cup SPATIAL \cup TIME\right)\right)^{+} \times instant\right)$ | STUPLE |
| stream | *STUPLE* | STREAM |
| now, unbounded, past | | WINDOW |
| srelation | $STREAM \times WINDOW$ | IRELATION |
| rstream | *IRELATION* | STREAM |
| streaming | $BASE \cup SPATIAL$ | TEMPORAL |

Budući da je sustav tipova podataka opisan signaturama, sorte u njemu u stvari su vrste.

Kao osnovu sustava tipova podataka uzimamo jednostavne tipove podataka. U definiranom sustavu tipova podataka, sadržani su u vrstama BASE, SPATIAL i TIME. Budući da nad njima gradimo sustav tipova podataka za cijelo jezero podataka, ključno je da svi uključeni sustavi pohrane i obrade podataka nude potporu za njih. Time se osigurava jednaka reprezentacija sustava tipova podataka u sloju obrade podataka i u sloju pohrane podataka.

U radovima [33], [34], [35] [36] opisana je semantika vrsta RANGE i TEMPORAL. Navedene vrste sadrže konstruktore tipova pomoću kojih se definiraju prostorno-vremenski objekti, bazirani na jednostavnim tipovima podataka. U odnosu na dosadašnje radove, pruža se

potpora za pomicanje nestrukturiranih tipova podataka (*blob*) u vremenu i prostoru primjenom odgovarajućih konstruktora. Primjerice, aplikacija konstruktora *moving* na tip podataka *blob* rezultira novim tipom *moving(blob)*, kojim opisujemo promjene stanja nestrukturiranog podatka kroz vrijeme. Tako se, primjerice, mogu pohranjivati nestrukturirani zapisi sa senzora, pri čemu obradu nestrukturiranog zapisa obavlja aplikacijski sloj izvan samog sustava pohrane.

Relacijska algebra temelji se na vrstama IDENTIFIER, TUPLE i RELATION. Korištenjem navedenih vrsta ostvaruje se potpora za pohranu prostorno-vremenskih tipova podataka u postojeće sustave za upravljanje bazama podataka. No, uz to, definira se i način upravljanja nestrukturiranim tipovima podataka unutar postrelacijskih baza podataka. Jedan od pristupa jest pohrana nestrukturiranog zapisa direktno u samu bazu podataka. Na taj način pruža se potpora zapisivanju u prirodnom obliku, najčešće kao nizu bajtova, uz rizik naglog povećanja korištenog prostora za pohranu. Drugi pristup uključuje pohranu zapisa unutar prostora u kojem je moguć veći stupanj kompresije podataka. U tom slučaju u bazu podataka pohranjuje se poveznica na podatak pohranjen unutar podatkovnog prostora.

Relacijske tipove podataka u kontekstu jezera podataka potrebno je promatrati u širem smislu od samih relacija. Pored postrelacijskih baza podataka, u kojima su ovi tipovi implementirani, podatke pohranjujemo u nestrukturiranim i polustrukturiranim oblicima. Polustrukturiranim podacima prilikom pohrane uobičajeno se pridodaje identifikator, što ih čini kandidatima za rad s relacijskim tipovima podataka. No, dio polustrukturiranih podataka sadrži ugniježđene dokumente s teoretski neograničenim dubinama, čime ne zadovoljavaju klasičnu formu ntorke. Strogo gledajući sa strane pohrane, polustrukturirani podaci su serijalizirani, što ih svrstava među ntorke spajajući vrste IDENTIFIER i BASE. Hijerarhija pritom dolazi do izražaja tek u sloju obrade podataka, gdje se problematika neograničene dubine rješava rekurzivnim funkcijama.

Stoga, zapise u polustrukturiranom i nestrukturiranom obliku promatramo kao skup parova zapisa oblika <K, V>. K pritom predstavlja naziv atributa, odnosno poopćeno ključ u paru zapisa, a V vrijednost atributa. Pojedini zapis predstavlja jednu ntorku, odnosno vrstu TUPLE. Skup ntorki, sukladno tome, predstavljen je vrstom RELATION.

Navedeni apstraktni sustav možemo iskazati signaturom drugog reda. Tada za signaturu $\Sigma = (K, \Gamma, T, \Delta, \Omega)$ vrijedi:

- $K = \{BASE,\ SPATIAL, TIME, RANGE,\ TEMPORAL,\ IDENTIFIER,\ TUPLE,\ RELATION,\ STUPLE,\ STREAM,\ WINDOWS,\ IRELATION\}$

- $\Gamma$ je $(S, \Sigma)$ gdje vrijedi $S = (K \cup T)$, $\Sigma = \{int, real, string, bool, blob, point, multiPoint, line, polygon, instant, range, moving, intime, tuple, relation, stuple, stream, now, unbounded, past, srelation, rstream, streaming\}$
- $T =$

$$\left\{ \begin{array}{c} int, real, string, bool, blob, \\ point, multiPoint, line, polygon, \\ instant, \\ range(int), range(real), range(string), range(bool), range(blob), range(instant), \\ moving(int), moving(real), moving(blob), moving(point), moving(multiPoint), \dots \\ intime(int), intime(real), intime(blob), intime(point), intime(multiPoint), \dots \\ id, \\ tuple(\dots), relation(tuple), stuple(\dots), stream(stuple), \\ now, unbounded, past, \\ srelation(\dots), \\ rstream(\dots), \\ streaming(int), streaming(real), streaming(blob), streaming(point), \dots \end{array} \right\}$$

### 3.1.3. Tip podataka *blob*

Jezera podataka oslanjaju se pretežno na pohranu nestrukturiranih podataka u njihovom izvornom obliku. Iz tog razloga, u sustavu podataka potrebno je specificirati tip podataka *blob* koji može pohraniti bilo koji nestrukturirani podatak. Sam tip podataka definiran je u SQL:1999 standardu [39], no uz sljedeća ograničenja:

- *blob* vrijednosti ne mogu biti primarni, alternativni ni strani ključevi
- podržane su operacije jednakosti i nejednakosti
- *blob* vrijednosti ne mogu biti dio GROUP BY i ORDER BY klauzula upita

U praksi, *blob* tip podatka rijetko je korišten u relacijskim bazama podataka. Njegov praktični doprinos bio je pohrana objekata u svrhu naknadne obrade, a kao zamjena pohrane datoteka u datotečnom sustavu. Uz to, metapodaci su pritom izostavljeni iz samog zapisa objekta te je za njihovu pohranu potrebno osigurati dodatne atribute unutar entiteta.

Unutar jezera podataka tip podataka *blob* došao je do punog izražaja, jer je osnovna premisa jezera podataka pohrana podataka u izvornom obliku. Promjenom paradigme iz korištenja upita nad strukturiranim podacima u obradu nestrukturiranih podataka, ograničenja koja su proizašla iz SQL standarda su neprimjenjiva. Sa strane same obrade podataka, tip podataka *blob* moguće

je obraditi u raspodijeljenom sustavu korištenjem paralelnih obrada te omogućava primjenu funkcija i procedura prilagođenih podacima koji strukturu ostvaruju u trenutku obrade.

Kao što je ranije navedeno, apliciranjem konstruktora tipa podatka koji pripadaju vrstama *temporal* i *range* nad konstruktorom tipa *blob* opisan je pomak nestrukturiranog podatka u prostoru odnosno vremenu. Tako dobivamo konstruktore tipa

- range (*blob*)
- moving (*blob*)
- intime (*blob*)

Važno je istaknuti kako nestrukturirani podatak može u sebi sadržavati eksplicitne vremenske i prostorne podatke. Primjerice, u slučaju nepromijenjenih podataka iz pokretnih senzora, niz bajtova dolazi kao jedan *blob* koji u konačnici može predstavljati više različitih podataka, no koji nisu eksplicitno prikazani u originalnom zapisu. Jedna od značajki nestrukturiranih podataka jest da u konačnici poprimaju određenu strukturu, ovisno o obradi podataka. Na taj se način dobivaju informacije iz deriviranih podataka kroz cjevovod obrade do njihove konačne strukture.

Kao primjer može se uzeti praćenje slikovnih zapisa iz nadzorne kamere. Takvi zapisi dolaze s vremenskom komponentom (vremenska oznaka trenutka u kojem je slikovni zapis nastao), prostornom komponentom (GPS koordinate uređaja) i nizom bajtova koji predstavljaju slikovni zapis.

Tip podataka *blob* predstavlja neprekinuti niz bajtova. Tako je njegova sintaksa vrlo bliska sintaksi tipa podataka *string*, iz razloga što u sebi može sadržavati bilo koji podržani znak. Važno je također istaknuti kako podatak tipa *blob* ne može nikada poprimiti vrijednost $\perp$, već nužno mora biti neprazni skup znakova.

Stoga, skup nosilac za tip podataka *blob* definira se na sljedeći način:

- Neka skup Sl sadrži mala i velika slova, $Sl = \{a - z\} \cup \{A - Z\}$
- Neka skup Br sadrži brojke od 0 do 9, $Br = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- Neka skup Zn sadrži sve posebne znakove, $Zn = \{\ , !, ", \#, \$, \%, \&, ', (, ), *, +, ,, -, ., /, :, ;, <, =, >, ?, @, [, \, ], \hat{}, \_, `, \{, |, \}, \sim, \ldots\}$
- Tada je skup nosilac za tip podataka *blob* $A_{blob} = (Sl \cup Br \cup Zn)^*$

Važno je istaknuti kako je gornja definicija skupa nosioca za tip podataka *blob* ograničena na tekstualne zapise kao reprezentaciju navedenog tipa podataka.

### 3.1.4. Operatori nad tipovima podataka

U radovima [32], [33] i [34] definirani su prethodno skupovi operatora nad tipovima podataka, osim nad tipom podataka *blob*. Iz tog razloga, potrebno je dopuniti postojeće skupove operatora novim operatorima kako bi konačni skup operatora bio sukladan tipovima podataka definiranima ranije u ovom poglavlju i prikazanima u tablici Tablica 1.

Ranije u poglavlju definirano je korištenje signature drugog reda Σ, temeljeno na radu [31]. Također, u radu [32] definiran je pojam proširene signature i njezina svrha. U signaturi drugog reda definirani su operatori nad tipovima podataka, kao što su operatori oblika *relation-to-relation,* sadržani u proširenoj signaturi $\Delta$. Proširena signatura $\Omega$ pak sadrži klasične operatore koji se ne odnose na same tipove podataka, kao što su operatori jednakosti, zbrajanja i sl..

Kada se radi o relacijskim operatorima, njihov skup je potpuno definiran u [32]. Relacije su pritom temeljene na strukturiranim podacima, dok je za polustrukturirane i nestrukturirane podatke potrebno definirati posebni skup operatora koji podržava njihovu strukturu. Budući da su u [32] definirani operatori oblika *relation-to-relation, relation-to-stream* i *stream-to-relation*, oni neće biti obuhvaćeni u ovom radu, već će operatori biti definirani za polustrukturirane i nestrukturirane podatke.

U ovom radu definirat će se operatori oblika *stream-to-document i document-to-document.* Operator oblika *document-to-stream* nije obuhvaćen definicijom, iz razloga što u kontekstu obrade podataka i njihove pohrane u jezera podataka nije primjenjiv. U slučajevima kada je jezero podataka izvor za daljnju obradu podataka, podaci se u povezane cjevovode ne dostavljaju u obliku tokova podataka. Operatori oblika *document-to-relation* i *relation-to-document* nisu obuhvaćeni ovom definicijom, budući da se radi o operacijama preslikavanja između dvije disjunktne strukture (ntorki i LPAV), što nije u fokusu istraživanja ovog rada. Iznimno, bit će eksplicitno definiran operator stvaranja sheme iz relacije, budući da se dio algoritama oslanja na njega.

Svi navedeni operatori odnose se na sve tipove podataka definirane u poglavlju 3.1.2.

Također, dio operatora specifičnih za LPAV definiran je u poglavlju 2.3.

**Operatori dokumenata (document-to-document)**

U ovom poglavlju definirani su operatori dokumenata u okviru proširene signature za opis operatora.

**Definicija 3.** Operator selekcije definiran je na sljedeći način:

$\forall dokument(LPAV)\ \boldsymbol{u}\ DOKUMENT$

$$dokument\ \times (f: LPAV \rightarrow bool)\ \rightarrow dokument\ (LPAV)$$

Operator selekcije, kao i u slučaju operatora selekcije u okviru relacijskih operatora, kao argument prima dokument i funkciju koja za LPAV iste strukture kao i zadani dokument vraća logičku vrijednost istina ili laž. Rezultat operacije selekcije jest dokument jednake strukture kao i dokument zadan u argumentu.

**Definicija 4.** Operator projekcije

Za zadani dokument $S = \{(a_1, v_1), \dots, (a_n, v_n)\}$ sa skupom atributa $\{a_1, \dots, a_n\}$ i skupom atributa $\{a_1, \dots, a_m\} \subseteq \{a_1, \dots, a_n\}$, pri čemu je $m \leq n$

Operator projekcije definiran je na sljedeći način:

$\forall dokument(\{(a_1, v_1), \dots, (a_n, v_n)\})\ \boldsymbol{u}\ DOKUMENT$

$$dokument\ \times \{a_1, \dots, a_m\}\ \rightarrow dokument(\{(a_1, v_m), \dots, (a_n, v_m)\})$$

Operator projekcije kao argument prima dokument i podskup atributa LPAV. Rezultat operacije projekcije je novi dokument čiji je skup atributa jednak skupu atributa zadanom kao argument operatora.

Operatori spajanja slijede definiciju operatora spajanja objekata i operatora spajanja klasa prikazanu u poglavlju 2.3., stoga u ovom poglavlju nije potrebna dodatna formalna specifikacija.

**Operatori *stream-to-document***

Operatori oblika stream-to-document odnose se na polustrukturirane oblike podataka, odnosno LPAV, koji pristižu u tokovima podataka te ih se pretvara u skup LPAV $S_1, \dots, S_n$. Tokove podataka temeljene na LPAV označavamo sa $\mathcal{S}$.

Navedeni operatori naslanjaju se na operatore klizećeg prozora definirane u [32], gdje su definirani operatori klizećih prozora za rad sa strukturiranim tokovima podataka temeljenima

na relacijama. Definicije operatora iz [32] prilagođene su radu s LPAV, uz zadržavanje definicija koje nisu eksplicitno vezane za relacije.

Na temelju definicije 25. iz [32], prilagođava se funkcija last na sljedeći način:

**Definicija 5.** Neka je za svaki tok podataka definirana funkcija *last* koja vraća indeks zadnjeg LPAV pristiglog u taj tok podataka.

Za svaki atribut toka podataka $a \in R$, $a_{last}$ označava vrijednost tog atributa u zadnjoj pristigloj LPAV.

Definicija 26. iz [32] prilagođava se na sljedeći način:

**Definicija 6.** Prozor NOW nad tokom podataka $\mathcal{S}$ je prozor utemeljen na vremenu koji u svakom trenutku sadrži LPAV koje su upravo u tom trenutku pristigli u tok podataka $\mathcal{S}$.

Definicija 27. iz [32] prilagođava se na sljedeći način:

**Definicija 7.** Prozor LAST nad tokom podataka $\mathcal{S}$ je prozor utemeljen na broju LPAV koji u svakom trenutku sadrži LPAV koja je posljednja pristigla u tok podataka $\mathcal{S}$.

Definicija 28. iz [32] prilagođava se na sljedeći način:

**Definicija 8.** Prozor UNBOUNDED nad tokom podataka $\mathcal{S}$ je prozor utemeljen na broju LPAV koji u svakom trenutku sadrži sve LPAV pristigle u tok podataka $\mathcal{S}$.

U sklopu proširene signature Ω, prozori NOW, LAST i UNBOUNDED definiraju se na sljedeći način:

$stream\ (S) \rightarrow document\ (S)\ \ NOW$

$stream\ (S) \rightarrow document\ (S)\ \ LAST$

$stream\ (S) \rightarrow document\ (S)\ \ UNBOUNDED$

Definicija 29. iz [32] prilagođava se na sljedeći način:

**Definicija 9.** Prozor PAST nad tokom podataka $\mathcal{S}$ je prozor temeljen na vremenu koji sadrži sve LPAV pristigle u tok podataka $\mathcal{S}$ u zadnjih $\Delta t$ jedinica vremena.

U sklopu proširene signature Ω, prozor PAST definira se na sljedeći način:

$stream\ (S) \times \Delta t\ \rightarrow document\ (S)\ \ PAST$

$stream\ (S) \times \Delta t \times slide \rightarrow document\ (S)\ \ PAST$

Definicija 30. iz [32] prilagođava se na sljedeći način:

**Definicija 10.** Prozor $SINCE$ nad tokom podataka $\mathcal{S}$ je prozor temeljen na vremenu koji sadrži sve LPAV pristigle u tok podataka $\mathcal{S}$ nakon vremenskog trenutka $t_0$.

U sklopu proširene signature Ω, prozor $SINCE$ definira se na sljedeći način:

$stream\ (S) \times t_0\ \rightarrow shema(S)\ \ SINCE$

$stream\ (S) \times t_0 \times slide \rightarrow shema(S)\ \ SINCE$

Definicija 31. iz [32] prilagođava se na sljedeći način:

**Definicija 11.** Prozor $RANGE$ nad tokom podataka $\mathcal{S}$ je prozor temeljen na broju LPAV koji sadrži posljednjih $r$ LPAV koje su pristigle u tok podataka $\mathcal{S}$.

U sklopu proširene signature Ω, prozor $RANGE$ definira se na sljedeći način:

$stream\ (S) \times r\ \rightarrow document\ (S)\ \ RANGE$

$stream\ (S) \times r \times slide \rightarrow document\ (S)\ \ RANGE$

Definicija 32. iz [32] prilagođava se na sljedeći način:

**Definicija 12.** Ako shema $S$ toka podataka $\mathcal{S}(S)$ sadrži prostorni atribut, nad tokom podataka $\mathcal{S}(S)$ moguće je definirati prostorni prozor. Neka je $p \in S$ prostorni atribut koji je dio sheme od $\mathcal{S}$. Neka je $F(p_1, p_2)$ zadana prostorna funkcija koja vraća logičku vrijednost (tip podataka bool). Prostorni prozor možemo definirati na sljedeći način:

U sklopu proširene signature Ω prostorni se prozori definiraju na sljedeći način:

$stream\ (S) \times \Delta t\ \times (f{:}\, SPATIAL \times SPATIAL \rightarrow bool) \rightarrow document\ (S)\ \ PAST$

$stream\ (S) \times \Delta t \times slide \times (f{:}\, SPATIAL \times SPATIAL \rightarrow bool) \rightarrow document\ (S)\ \ PAST$

$stream\ (S) \times r\ \times (f{:}\, SPATIAL \times SPATIAL \rightarrow bool) \rightarrow document\ (S)\ \ SP\_RANGE$

$stream\ (S) \times r \times slide \times (f{:}\, SPATIAL \times SPATIAL \rightarrow bool)$
$\rightarrow document\ (S)\ \ SP\_RANGE$

Definicija 33. iz [32] prilagođava se na sljedeći način:

**Definicija 13.** Neka je $\mathcal{S}$ tok podataka nad shemom $S$. Neka je $A$ podskup atributa sheme $S$. S $Part\ (\mathcal{S}) = \{\mathcal{S}_{A1}, \mathcal{S}_{A2}, \dots, \mathcal{S}_{Ap}\}$ označavamo particionirani skup toka $\mathcal{S}$ koji se sastoji od podskupova od $\mathcal{S}$ s jednakim vrijednostima atributa podskupa atributa $A$.

Particionirani prozori definirani su tada kao:

$$LASTP(\mathcal{S}, A) = \bigcup_{j=1}^{p} \quad LAST(\mathcal{S}_{Aj})$$

$$RANGEP(\mathcal{S}, r, A) = \bigcup_{j=1}^{p} \quad RANGE(\mathcal{S}_{Aj}, r)$$

$$SP\_RANGEP(\mathcal{S}, r, F, A) = \bigcup_{j=1}^{p} \quad SP\_RANGE(\mathcal{S}_{Aj}, r, F)$$

Sintaksa operatora particioniranog klizećeg prozora u sklopu proširene signature Ω dana je u tablici Tablica 2.

Tablica 2. Particionirani prozori

| Operator | Tipovi argumenata | Tip rezultata |
|---|---|---|
| $LASTP$ | $stream\ (S) \times \{id^+\}$ | $\rightarrow document\ (S)$ |
| $RANGEP$ | $stream\ (S) \times int\ \times \{id^+\}$ | $\rightarrow document\ (S)$ |
| $SP\_RANGEP$ | $stream\ (S) \times int\ \times \{id^+\}$ $\times\ (f: SPATIAL \times SPATIAL \rightarrow bool)$ | $\rightarrow document\ (S)$ |

Kao što je vidljivo iz prilagođenih definicija, operatori oblika *stream-to-document* svojim ponašanjem analogni su operatorima oblika *stream-to-relation*. Razlika dolazi u implementaciji gdje SUBP, umjesto rada s ntorkama, obrađuje LPAV. No, u kontekstu signature drugog reda i proširene signature, u formalnoj definiciji nema značajnih razlika.

**Operatori relation-to-document**

Kako je ranije navedeno, definiramo isključivo operator stvaranja dokumenta iz relacije, koji je nužan za kasnije definicije algoritama.

**Definicija 14.** Neka je zadana relacija s relacijskom shemom

$R = < \{(id_1, v_1), \dots, (id_n, v_n), \} >$, pri čemu $id_1, \dots, id_n$ predstavljaju atribute relacije

Relaciji $R$ istovjetan je LPAV

$S = \{(a_1, v_1), \dots, (a_n, v_n)\}$

sa skupom atributa $\{a_1, \dots, a_n\}$

## 3.2. Prilagodba modela jezera podataka

### 3.2.1. Prijedlog logičkog modela jezera podataka

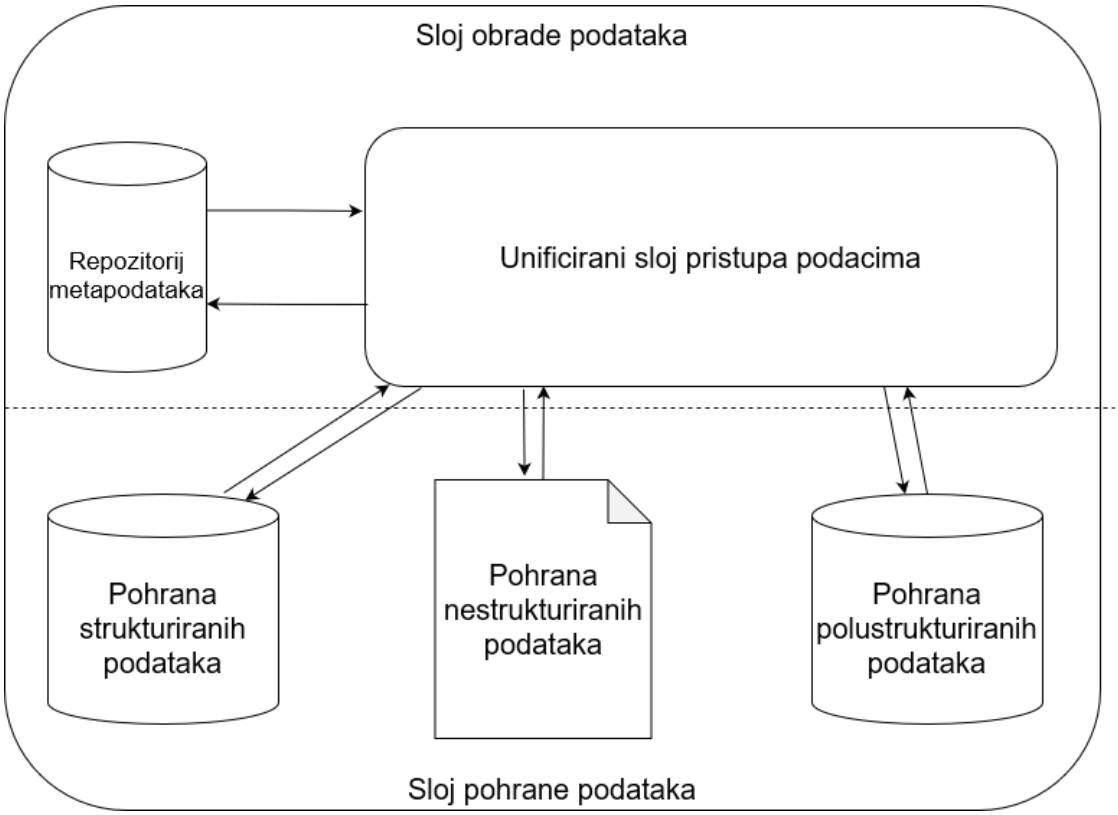
Predloženi prilagođeni model jezera podataka u konačnici definiran je u dvije dimenzije – u slojevima i u zonama. Slojevi logički razdvajaju funkcionalne cjeline jezera podataka:

- Cjelina obrade i pristupa, koja neovisno o konačnom obliku podatke prihvaća, prosljeđuje, po potrebi preoblikuje (transformira) i izlaže u obliku pogodnom za pohranu u perzistentnu memoriju ili daljnje korištenje izvan jezera podataka. U cjelini obrade i pristupa nalaze se podatkovni cjevovodi kojima su ostvarene tražene zadaće.
- Cjelina pohrane, koja podržava pohranu strukturiranih, polustrukturiranih i nestrukturiranih podataka u obliku dobivenom kroz procese obrade iz cjeline obrade. Podaci se u cjelini pohrane zapisuju trajno u perzistentnu memoriju i uglavnom nisu namijenjeni brisanju. Također, cjelina pohrane omogućava dohvat pohranjenih podataka cjelini obrade u svrhu dodatne obrade postojećih podataka ili za potrebe korištenja u vanjskim sustavima, poput analitičkih alata.

Kroz cjelinu obrade i pristupa podacima istovremeno su definirane zone jezera podataka, budući da se one naslanjaju na pohranjene podatke. Pojedina zona jezera podataka može konzumirati podatke ovisno o potrebnim oblicima, stoga je nužno da svim zonama budu istovremeno dostupni svi oblici pohrane podataka. Unutar cjeline obrade i pristupa podacima, zone jezera podataka predstavljaju logičku podjelu nad organizacijom prihvata, pohrane i izlaganja podataka. Same zone također utječu na zadaće i kompleksnost pojedinih podatkovnih cjevovoda.

Zone jezera podataka, kao logičke organizacijske cjeline, u potpunosti su definirane metapodacima. Pojedina zona sadržava entitete koji potječu iz samih izvora podataka ili iz drugih zona kao posljedica obavljanja operacija preslikavanja ili pretvorbe.

U predloženom modelu jezera podataka navedene dvije logičke cjeline nazvane su sloj obrade podataka i sloj pohrane podataka, s podcjelinama kako je prikazano na slici 11. Sama struktura podjele direktno ocrtava njihove namjene, pri čemu sloj obrade podataka ima dva neovisna skupa sučelja - vanjska i unutarnja, dok su u sloju pohrane podataka sučelja definirana strukturom pohrane podataka, odnosno njihovim vlastitim sustavom za upravljanje.




Slika 11. Arhitektura predloženog modela jezera podataka

Na vanjskim sučeljima sloja obrade podataka obavljaju se zadaće čitanja podataka na pripadnim sučeljima, prilagođenima strukturi dolaznih podataka te izlaganja podataka u prilagođenim oblicima za dohvat vanjskim sustavima. Unutarnja sučelja, analogno, imaju zadaću izlaganja podatka prema sustavima za pohranu podataka te prihvata podataka iz njih.

Sloj pohrane podataka za svaki oblik pohrane podataka traži zasebno sučelje koje komunicira dvosmjerno putem skupa naredbi, procedura i funkcija definiranim u pojedinom sustavu za pohranu podataka. Podaci na sučelja dolaze u obliku prikladnom za pojedino sučelje, za što je zadužen sloj pristupa podacima kroz svoja unutarnja sučelja, a ujedno i šalje podatke prema unutarnjim sučeljima u obliku definiranom u pojedinom sustavu za pohranu podataka. Za svaki sustav pohrane podataka potrebna su minimalno dva unutarnja sučelja u sloju pristupa podacima – barem po jedno putem kojeg sustav prima podatke te po jedno putem kojeg sustav izlaže podatke.

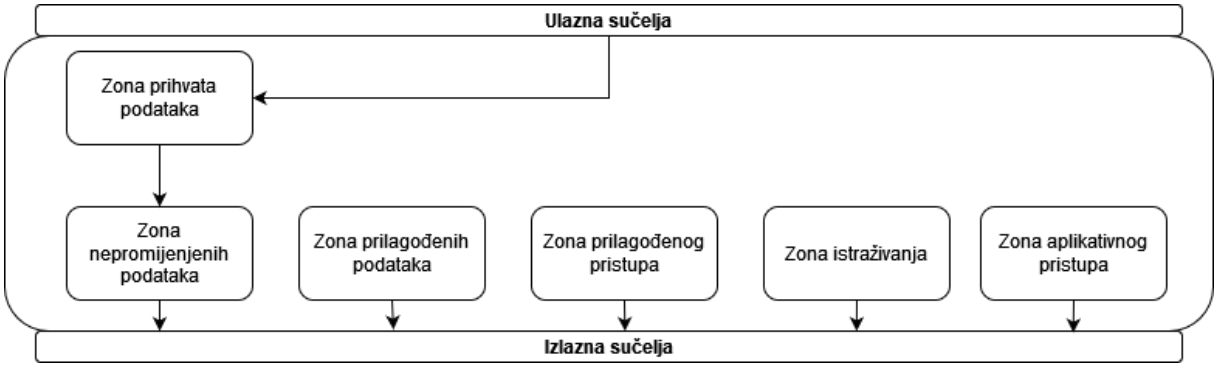
Model jezera podataka prema vanjskim sustavima predstavlja se kao jedinstvena cjelina, pri čemu su poznati načini pristupa podacima, podržani tipovi podataka i operacije. Vrste podataka i operacije definiramo na jedinstveni način na razini cijelog modela jezera podataka, čime se daje čvrsti temelj za unificirani pristup podacima i njihovu obradu. Iz tog razloga, uz prijedlog

modela arhitekture jezera podataka formalno se specificiraju svi podržani tipovi podataka i operacije nad njima koji moraju biti podržani u svakoj od komponenti jezera podataka. Time se u određenoj mjeri i ograničava skup tehnoloških rješenja pomoću kojih je moguća implementacija samog jezera podataka.

Zonska arhitektura u predloženom jezeru podataka, osim u dijelu prihvata podataka, u potpunosti koristi poveznicu sa slojem pohrane podataka. Zone podatke koje imaju funkciju dohvata podatke preuzimaju iz sloja pohrane podataka, gdje su oni pohranjeni u obliku koji logički i strukturom odgovara namjeni pojedine zone. Zona nepromijenjenih podataka tako koristi dostupna sučelja prema sloju pohrane podataka, pomoću kojih dohvaća podatke pohranjene nakon završenog procesa gutanja podataka. Važno je naglasiti kako predložene zone mogu konzumirati podatke tek nakon pohrane u perzistentnu memoriju. Podaci u procesu obrade nisu dostupni za rad, osim u privremenoj memoriji koja se nalazi u sloju obrade podataka i koje vanjski sustavi mogu koristiti za privremene analize u obliku brzih podataka.

Predloženi model jezera podataka podijeljen je u šest zona, kako je vidljivo na slici 12:

- Zona prihvata podataka
- Zona nepromijenjenih podataka
- Zona prilagođenih podataka
- Zona prilagođenog pristupa
- Zona istraživanja
- Zona aplikativnog pristupa




Slika 12. Zonska podjela jezera podataka

**Zona prihvata podataka**

U zoni prihvata podataka nalaze se podaci dohvaćeni iz vanjskih sustava putem odgovarajućih sučelja. U njoj podaci ostaju do trenutka pohrane u perzistentnu memoriju,

odnosno postanu dostupni u zoni nepromijenjenih podataka. Za čišćenje zone prihvata podataka od pohranjenih podataka zaduženi su odgovarajući mehanizmi koji su ugrađeni u sloj obrade podataka.

Zona prihvata podataka jedina je od zona koja nema direktnu komunikaciju s cjelinom pohrane podataka te se jedina ne odnosi na logičku organizaciju podataka u perzistentnoj memoriji. Za podatke u zoni prihvata podataka nužno je postojanje privremenog spremnika podataka, gdje su podaci pohranjeni i dostupni za obradu kroz cjevovode, odnosno konačnu pohranu u perzistentnu memoriju.

**Zona nepromijenjenih podataka**

Zona nepromijenjenih podataka sadrži podatke pohranjene u njihovom izvornom, nepromijenjenom obliku korištenjem procesa gutanja podataka. U ovoj zoni podaci su dohvaćeni iz perzistentne memorije, korištenjem unutarnjih sučelja.

Podaci dohvaćeni preko vanjskih sučelja, nakon inicijalne obrade i gutanja podataka, prvo postaju dostupni u zoni nepromijenjenih podataka. U preostalim zonama, njihova dostupnost ovisi o kompleksnosti i postojanju naknadnih obrada.

Podaci u zoni nepromijenjenih podataka, u pravilu, nisu dostupni za dohvat putem aplikacija ili korisničkih upita. Zona nepromijenjenih podataka prvenstveno ima zadaću osigurati idempotentni dohvat podataka nakon same pohrane kako bi oni kroz ostale zone postali dostupni, bilo u nepromijenjenom, bilo u prilagođenom obliku.

**Zona prilagođenih podataka**

U zoni prilagođenih podataka prisutni su podaci prilagođeni kroz procese integracije ili dorade u podatkovnim cjevovodima. Uloga zone prilagođenih podataka jest podatke iz perzistentne memorije prikazati u jednakoj strukturi unutar cijele zone.

Kao oblik strukture podataka unutar zone prilagođenih podataka predviđeno je korištenje polustrukturiranih zapisa. Prvenstveni razlog za odabir polustrukturiranih, umjesto strukturiranih zapisa, leži u činjenici da se nestrukturirani podaci značajno lakše preslikavaju u polustrukturirani oblik u odnosu na strukturirani. Također, strukturirani podaci se iznimno jednostavno mogu preslikati u polustrukturirani oblik, dok pretvaranje polustrukturiranih podataka u strukturirani oblik postaje kompleksno kod više razina hijerarhija podataka.

Nadalje, dio alata za naprednu analizu podataka omogućuju izvršavanje SQL upita nad polustrukturiranim podacima. Time se uklanja prepreka za obavljanje analitičkih upita kroz poznate upitne jezike.

**Zona prilagođenog pristupa**

Zona prilagođenog pristupa naslanja se na zonu prilagođenih podataka, no uvodi dodatni sloj kontrole razina pristupa. Pristup podacima u zoni prilagođenog pristupa moguće je definirati na više razina, od pojedinačnog atributa unutar pohranjenog entiteta, preko razine samog entiteta do skupa entiteta grupiranog u odgovarajuće subjektno područje.

Navedena zona velikim je dijelom analogna s područnim skladištem podataka, u kojem je pristup podacima ograničen unutar organizacijskih jedinica prema potrebama pojedinog korisnika ili grupe korisnika. Također, u zoni prilagođenog pristupa podaci su oblikovani prema potrebama krajnjih korisnika, ne samo u smislu strukture. To se odnosi na primjenu poslovnih pravila ako su potrebna te primjenu procesa upravljanja podacima, pri čemu su primarno primijenjeni procesi kontrole kvalitete podataka.

Razine pristupa podacima definiraju se u repozitoriju metapodataka kroz odgovarajuće klase.

Zona prilagođenog pristupa, kao i zona prilagođenih podataka, sadrži podatke u polustrukturiranom obliku.

**Zona istraživanja**

Zona istraživanja sadrži sve podatke pohranjene u perzistentnoj memoriji dostupne za korisničku analizu. Podaci koji su dostupni kroz zonu istraživanja čine uniju podataka iz zone nepromijenjenih podataka i zone prilagođenih podataka.

Uloga zone istraživanja je pružanje neograničenog pristupa svim podacima dostupnima u jezeru podataka u svrhu korisničkih upita i analiza, odnosno pružanja potpore zadacima napredne analize, dubinske analize podataka i strojnog učenja. Kako je za te zadatke nužan neometan pristup podacima, ostali slojevi zbog svojih zadaća tek su djelomično pogodni za navedene zadatke.

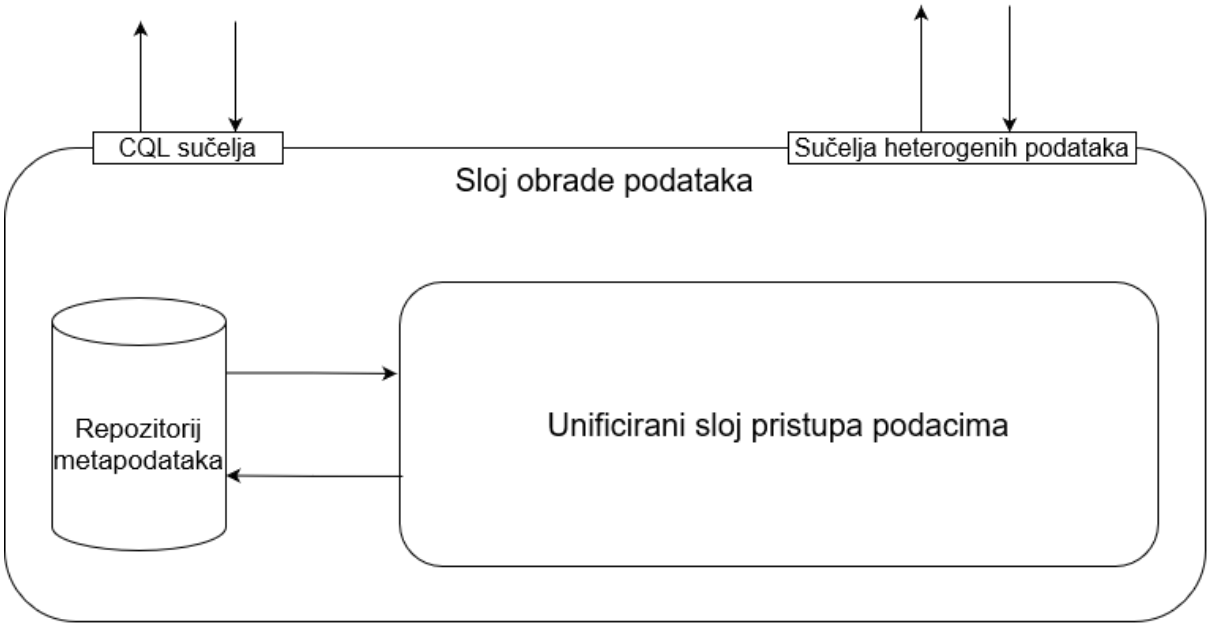
**Zona aplikativnog pristupa**

U zoni aplikativnog pristupa podaci su oblikovani prema potrebama poslovnih i analitičkih aplikacija, koje ih konzumiraju bilo u nepromijenjenom, bilo u prilagođenom obliku. Razlika ove zone od zone prilagođenog oblika, zone prilagođenog pristupa i zone nepromijenjenih

podataka leži u činjenici kako aplikacije ne obavljaju iznimno heterogene upite i zadatke nad podacima, već su oni često repetitivni i sličnog oblika. Iz tog su razloga podaci u zoni aplikativnog pristupa prethodno prilagođeni strukturom pojedinoj aplikaciji. Također, u aplikativnom pristupu pristup podacima ograničava se na razini pojedine aplikacije, a ne na razini korisnika.

### 3.3. Sloj obrade podataka

Uloga sloja obrade podataka jest prikupljanje, obrada i predstavljanje strukturiranih, polustrukturiranih i nestrukturiranih podataka pristiglih iz vanjskih sustava ili iz sloja pohrane podataka na unificiran način. Sastoji se od dvije glavne jedinice: repozitorija metapodataka pomoću kojeg se ostvaruje kvalitetno praćenje podataka kroz cijelo jezero podataka te unificiranog sloja pristupa podacima koji ima zadaću prilagodbe oblika dolaznih ili odlaznih podataka u prikladne strukture te usmjeravanje prema sučeljima pohrane podataka odnosno vanjskih sustava. Arhitektura sloja obrade podataka prikazana je na slici 13.




Slika 13. Sloj obrade podataka

Unificirani sloj pristupa podacima svoje procese temelji na podacima iz repozitorija metapodataka. Procesi obrade podataka u dinamičkom okruženju kao što je jezero podataka češće se mijenjaju od obrada podataka u okruženjima isključivo strukturiranih podataka. Iz tog razloga, dinamiku promjene izvora, strukture ili obrade podataka nužno je popratiti odgovarajućim zapisima u samom repozitoriju metapodataka.

Unificirani sloj pristupa podacima ostvaren je kroz niz podatkovnih cjevovoda, u kojima se podaci prenose iz izvora podataka prema njihovom odredištu, unutar ili izvan jezera podataka. Izvori podataka za unificirani sloj pristupa podacima mogu biti vanjski sustavi, ali i sustavi pohrane podataka unutar jezera podataka, koji omogućuju dodatnu obradu podataka pohranjenih u perzistentnu memoriju i stvaranje njihovih prilagođenih oblika za daljnju analizu.

Temelj komunikacije sloja obrade podataka s vanjskim okruženjem jezera podataka (izvornim sustavima, analitičkim aplikacijama i sl.) i slojem pohrane podataka su pristupna sučelja. Prvenstvena uloga pristupnih sučelja je ostvarivanje početnih i krajnjih točaka podatkovnih cjevovoda, budući da podaci mogu biti kroz cjevovode usmjereni i oblikovani prema potrebi.

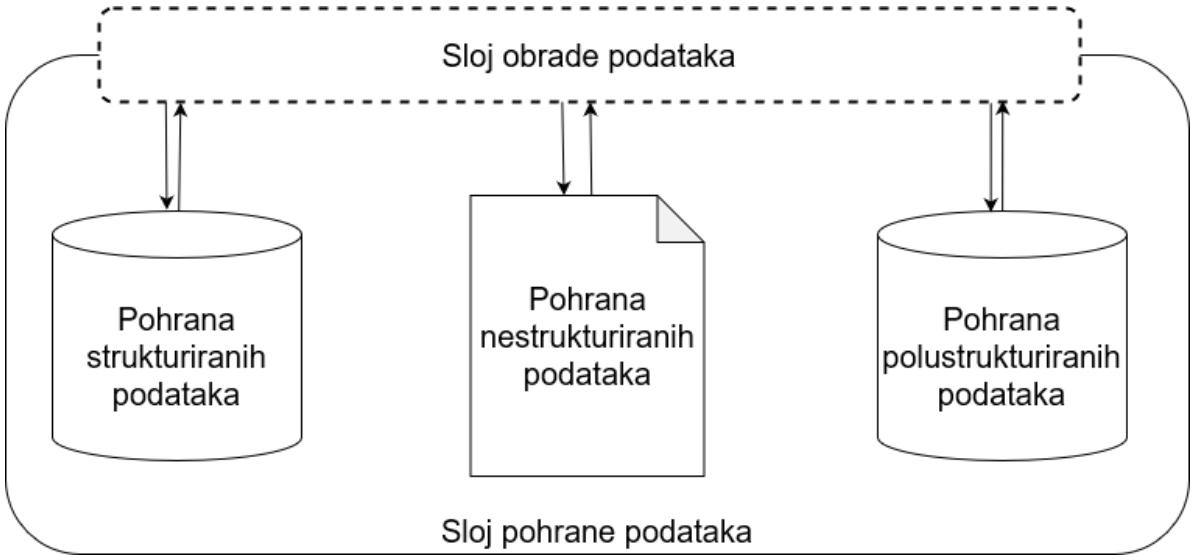
Sloj obrade podataka ne sadrži komponentu koja bi predstavljala perzistentne zapise prihvaćenih podataka. Pohrana podataka koja se odvija unutar sloja obrade podataka isključivo ima ulogu privremene pohrane do trenutka ostvarivanja perzistentnog zapisa unutar sloja pohrane podataka. Privremena pohrana podataka je pak osnova ostvarivanja potpore Lambda arhitekturi, točnije sloju brzih podataka kojeg ona definira. Također, privremena pohrana podataka direktno se oslanja na unificirani sloj pristupa podacima, a njezina implementacija ovisi o mehanizmima korištenima u njemu. Primjerice, alat za ostvarivanje modela objavi-pretplati poput Apache Kafka u svojoj implementaciji sadrži privremeni spremnik podataka koji time postaje dio sloja obrade podataka.

Pored navedenog, sloj obrade podataka ima ulogu virtualizacijskog sloja nad jezerom podataka. Zadaće virtualizacijskog sloja su višestruke. Prvenstveno, kroz virtualizacijski sloj sakriva se kompleksnost modela jezera podataka, obrade podataka te načina njihove pohrane. Korisnicima i aplikacijama potrebno je omogućiti pristup samo dijelovima jezera podataka koji su im nužni za rad, što se ostvaruje kroz zonsku arhitekturu i uspostavljanje procesa upravljanja podacima. Nadalje, virtualizacijski sloj omogućuje spajanje podataka iz više različitih sustava pohrane podataka kroz mehanizme spajanja u jedinstveni rezultat, prikazan u prethodno definiranoj strukturi. U tu svrhu nužan je uspostavljeni i ispravno održavan repozitorij metapodataka, budući da se pomoću metapodataka i definiranih pravila preslikavanja i spajanja heterogeni podaci mogu prikazati u jedinstvenoj strukturi za svaki pojedinačni upit.

### 3.4. Sloj pohrane podataka

U sloju pohrane podataka modela jezera podataka postoje tri cjeline, koje odgovaraju tri različita načina pohrane podataka, kako je prikazano na slici 14. Slijedom arhitekture repozitorija u kojoj postoje pohranjeni izvorni, nepromijenjeni podaci i podaci u prilagođenom obliku, model mora pružati potporu za pohranu strukturiranih, nestrukturiranih i polustrukturiranih podataka.

Iz tog razloga, podaci se usmjeravaju kroz sloj obrade podataka prema unutarnjim sučeljima ovisno o njihovoj izvornoj strukturi. Tako se osigurava mogućnost pohrane podataka bez promjene strukture zapisa, izvršavanje operacija prilagođenih pojedinog strukturi, ali i repliciranje podataka u drugim strukturama i pohrana u ostalim sustavima.



Slika 14. Arhitektura sloja pohrane podataka

Ovakav način definiranja pohrane igra važnu ulogu u ostvarivanju zonske arhitekture kroz sučelja. Unificirani sloj pristupa podacima, na temelju metapodataka, poznaje strukturu ulaznih podataka i sučelja za pohranu odgovarajućih struktura podataka te kroz podatkovne cjevovode može usmjeriti podatke u pripadno izlazno sučelje. Time se ostvaruje pohrana podataka u sustav koji je prilagođen njihovoj izvornoj strukturi.

Kako se u zonskoj arhitekturi isti podaci pohranjuju u različitim oblicima, ovisno o zoni za koju su prilagođeni, nužno je ostvariti istovremenu pohranu istih podataka u različitim strukturama, ali ujedno i poznavati koja je njihova izvorna struktura. Stoga se nepromijenjeni podaci pohranjuju prvi u sustav i postaju dostupni u zoni nepromijenjenih podataka.

Po pohrani podataka u zoni nepromijenjenih podataka, smatra se kako su u tom trenutku podaci došli u perzistentnu memoriju, aktivira se mehanizam za njihovo micanje iz sloja brzih

podataka, odnosno privremenog spremnika. Ujedno se kroz sučelja mogu ponovno dohvatiti u podatkovne cjevovode te postaju dostupni za daljnje obrade kojima se prilagođavaju za pohranu u obliku prilagođenom preostalim zonama.

Pohrana istih podataka u različitim vrstama strukturama posebno je kompleksna u zoni istraživanja, budući je u njoj omogućen pristup svim podacima pohranjenima u jezeru podataka. Pritom važnu ulogu ima repozitorij metapodataka, u kojem se nalaze podaci o promjenama strukture pojedinog entiteta. Na taj način moguća je osnovna identifikacija istovrsnih entiteta, čime se izbjegava njihovo potencijalno višestruko korištenje prilikom analize.

Važnu ulogu u pohrani prilagođenih podataka ima pohrana polustrukturiranih podataka. U tom dijelu sloja pohrane podataka prisutne su NoSQL baze podataka u svrhu potpore radu s polustrukturiranim podacima. Pritom se pohrana odnosi na dokument i graf NoSQL baze podataka. Uključivanje dodatnih vrsta NoSQL baza podataka omogućuje obavljanje specifičnih analiza nad podacima. Primjerice, graf baze podataka pružaju mogućnost jednostavne implementacije algoritama specifičnih za grafove, kao što je traženje najkraćeg puta između dva čvora. Na primjer, u kontekstu prostorno-vremenskih podataka, potreba za analizama usmjerenih putanja, koja se ostvaruje kroz analizu grafa, je vrlo logičan zahtjev, uzevši u obzir specifičnost oblika i namjene samih podataka.

## 3.5. Unificirani sloj pristupa podacima (Unified Data Access Layer - UDAL)

### 3.5.1. Definicija UDAL modela

Unificirani sloj pristupa podacima u svojoj je osnovi međusloj koji ima sljedeće osnovne zadaće:

- prihvat podataka iz vanjskih izvora pomoću sučelja
- stvaranje podatkovnih cjevovoda u svrhu prijenosa prihvaćenih podataka između sučelja
- izlaganje podataka prema sustavima za pohranu podataka

Skup zadaća unificiranog sloja pristupa podacima može biti proširen u svrhu ostvarivanja dodatnih mogućnosti. Primjer takvih zadaća su uspostavljanje nadzora kvalitete podataka, rana detekcija sličnosti entiteta ili uspostavljanje virtualizacijskog sloja nad podacima.

Unificirani sloj pristupa podacima, označen s UDAL, sastoji se od skupa ulaznih sučelja ($\mathcal{S}_u$), skupa upravitelja podataka ($\mathcal{U}$) i skupa izlaznih sučelja ($\mathcal{S}_i$). Formalno definirano, unificirani sloj pristup podacima može se iskazati na sljedeći način:

$$USPP = \begin{cases} \mathcal{S}_{ui}, 1 \leq i \leq m, pri\ čemu\ je\ m\ ukupni\ broj\ uzlaznih\ sučelja \\ \mathcal{M}_j, 1 \leq j \leq n, pri\ čemu\ je\ n\ ukupni\ broj\ upravitelja \\ \mathcal{S}_{ik}, 1 \leq k \leq o, pri\ čemu\ je\ o\ ukupni\ broj\ izlaznih\ sučelja \end{cases}$$

Podatkovni cjevovod, označen s $PC$, sastoji se od minimalno jednog ulaznog sučelja, minimalno jednog upravitelja uz pripadne procedure, iskazanih kao par $< \mathcal{M}, \mathcal{p} >$, i minimalno jednog izlaznog sučelja, odnosno

$$PC = \left\{\{\mathcal{S}_{u1}, \dots, \mathcal{S}_{up}\}, \{< \mathcal{M}_1, \mathcal{p}_1 >, \dots, < \mathcal{M}_q, \mathcal{p}_r >\}, \{\mathcal{S}_{i1}, \dots, \mathcal{S}_{iq}\}\right\}$$

pri čemu podatkovni cjevovod sadrži $p \geq 1$ ulaznih sučelja, $r \geq 1$ upravitelja uz pripadne procedure i $q \geq 1$ izlaznih sučelja.

Unutar unificiranog sloja pristupa podacima nalaze se tri skupa podatkovnih cjevovoda, po jedan za svaku vrstu struktura podataka:

$$USPP = \{PC_S \cup PC_P \cup PC_N\}$$

$PC_S$ označava skup podatkovnih cjevovoda za strukturirane, $PC_P$ za polustrukturirane i $PC_N$ za nestrukturirane podatke.

Svi podatkovni cjevovodi međusobno su neovisni, iako mogu dijeliti elemente i koristiti iste procedure u pojedinim procesima. Razlika među podatkovnim cjevovodima najčešće se nalazi u dijelu podatkovnog cjevovoda zaduženom za gutanje, odnosno integraciju podataka.

Jedan podatkovni cjevovod može prihvaćati podatke iz više sučelja, koja mogu biti heterogene strukture. U tom slučaju, podatkovni cjevovod nikako ne može sadržavati samo proces gutanja podataka, već se radi o procesu integracije. Ujedno, za takve podatkovne cjevovode potrebna je eksplicitna klasifikacija strukturne pripadnosti prilikom uspostavljanja definicije u repozitoriju metapodataka. Pojedini podatkovni cjevovod može pripadati samo jednom strukturalnom skupu podatkovnih cjevovoda.

Unificirani sloj pristupa podacima u potpunosti je opisan u repozitoriju metapodataka u odgovarajućim entitetima do razine pojedinog atributa. Svakom elementu pojedinog podatkovnog cjevovoda pripada jedan zapis za pripadni rezultirajući entitet.

### 3.5.2. Ulazna sučelja

Ulazna sučelja u predloženom modelu analogna su komponenti apstrakcijskog aplikacijskog programskog sučelja iz prethodno opisanog metamodela međusloja pristupa podacima. Namjena ulaznih sučelja u predloženom modelu unificiranog sloja pristupa podacima jest ostvarivanje prihvata podataka od strane izvora podataka.

Ulazna sučelja predstavljaju ulaznu točku podataka u podatkovni cjevovod. Izvori podataka imaju mogućnosti biti pretplaćeni na pojedino sučelje, odnosno pojedino sučelje može biti pretplaćeno na jedan ili više izvora podataka. U slučaju da više izvora podataka dijeli model podataka, pojedino sučelje može biti definirano kao dijeljena potpora za te izvore podataka. Poglavito se to odnosi na izvore podataka poznate strukture, odnosno - u slučaju polustrukturiranih podataka - sheme u kojima je, temeljem poznatih metapodataka izvora podataka, moguće optimizirati broj sučelja.

Specijalizacija ulaznih sučelja dovodi do prve osnovne podjele prema strukturi podataka. Tako ulazna sučelja mogu prihvaćati strukturirane, polustrukturirane i nestrukturirane podatke. Sučelja se također razlikuju prema namjeni potpore, pri čemu sučelje može biti specijalizirano za potporu serijama (*eng. batch*) podataka i potporu tokovima podataka. Pritom je važno naglasiti da ulazna sučelja za prihvat strukturiranih tokova podataka impliciraju potporu upitnom jeziku CRQ. Specijalizacija sučelja definira se u repozitoriju metapodataka.

Sama ulazna sučelja, iako mogu posjedovati mehanizam prihvata podataka, prvenstveno imaju ulogu predstavljanja strukture i lokacije dostave podataka. Zadaća upravljanja prihvatom podataka ostvaruje se uvođenjem upravitelja koji komunicira s odgovarajućim ulaznim sučeljima. Prihvat podataka na razini ulaznog sučelja ograničen je na privremenu pohranu posljednjeg dostavljenog skupa podataka i mehanizam dojave o promjenama nad podacima. Ulazna se sučelja stvaraju po završetku unosa njihove definicije u repozitorij metapodataka pomoću automatiziranih procedura.

### 3.5.3. Izlazna sučelja

Izlazna sučelja predstavljaju prvu točku pristupa pohrani podataka u unificiranom sloju pristupa podacima. Djelomično odgovaraju komponenti sučelja spajanja na pohranu podataka - u dijelu pohrane podataka u perzistentnu memoriju. Izlazna sučelja, za razliku od ulaznih sučelja, striktno su vezana uz sloj pohrane podataka, gdje je svakom obliku pohrane podataka pridodano njegovo sučelje. Sustavi za pohranu podataka dolaze s vlastitim upravljačkim programima te se oni pripajaju pojedinom izlaznom sučelju, čime se ostvaruje komunikacija prema sustavu za upravljanje pohranom podataka.

U kontekstu kompleksnijeg sustava, kao što je jezero podataka s potporom za rad s tokovima podataka, izlazna sučelja trebaju pružati potporu za Lambda arhitekturu. U tu svrhu potrebno je izložiti neobrađene podatke kroz sučelja kako bi odgovarajući alati imali pristup podacima u brzom sloju.

Izlazna sučelja imaju višestruku ulogu u unificiranom sloju pristupa podacima. Pored izlaganja podataka sloju pohrane podataka, omogućuju vanjskim sustavima pristup podacima pohranjenima u zonama jezera podataka.

S gledišta Lambda arhitekture, potrebno je osigurati mogućnost pristupa podacima koji su učitani u unificirani sloj pristupa podacima, no još nisu preoblikovani ni pohranjeni u perzistentnu memoriju. Navedeni podatkovni cjevovod, koji odgovara sloju brzih podataka u Lambda arhitekturi, po svojoj prirodi obrađuje nepromijenjene izvorne podatke i radi nad manjim brojem podataka. Stoga je unutar unificiranog sloja pristupa podacima potrebno osigurati sučelja koja omogućuju dohvat i obradu takvih podataka. Takva sučelja nazivaju se sučelja izvornih podataka i omogućuju aplikacijama za naprednu analizu podataka rad nad neobrađenim podacima.

Budući da postoji određeni vremenski odmak od trenutka dohvata podataka iz unificiranog sloja pristupa podacima do njihovog stvarnog zapisa u perzistentnu memoriju, pomoću sučelja izvornih podataka ostvaruje se potpora gotovo stvarnovremenskoj analizi podataka.

Uz to, potpuna potpora Lambda arhitekturi zahtjeva kombinaciju gotovo stvarnovremenskih podataka i podataka iz perzistentne memorije. Stoga je nužno u tu svrhu istovremeno omogućiti pristup obrađenim podacima u perzistentnoj memoriji putem dodatnog skupa specijaliziranih sučelja. Navedena sučelja proširuju pristup obrađenim podacima, koji nisu nužno i izvorni pohranjeni podaci. U pravilu, to se odnosi na zone harmoniziranih i pročišćenih podatka u jezeru podataka, no ujedno uključuju i podatke iz zona isporuke i istraživanja. Specijalizirana sučelja se pritom značajno oslanjaju na repozitorij metapodataka, u kojem su definirane strukture pohranjenih podataka, kako bi korisnicima pružili mogućnost analiza. Valja naglasiti da specijalizirana sučelja ujedno koriste i neovisne podatkovne cjevovode.

Zajedno, sučelja izvornih podataka i specijalizirana sučelja logički tvore uslužni sloj Lambda arhitekture, koji je potom izložen vanjskim aplikacijama. Valja istaknuti kako zona nepromijenjenih podataka nije izložena putem sučelja, zato što su podaci pohranjeni u njoj sadržani u preostalim zonama u izvornom i obrađenom obliku.

### 3.5.4. Upravitelji

Uloga upravitelja u unificiranom sloju pristupa podacima jest nadzor i usmjeravanje podataka unutar pojedinih podatkovnih cjevovoda koji su definirani ulaznim i izlaznim sučeljima. Pojedini podatkovni cjevovod može sadržavati jedan ili više upravitelja, ovisno o kompleksnosti potrebnih operacije i samoj namjeni cjevovoda.

Generalno gledajući, upravitelji imaju procesnu ulogu u jedinstvenom sloju pristupa podacima. Sam jedinstveni sloj ima više različitih procesa koji se odvijaju istovremeno, ponekad obavljajući različite operacije nad istim skupovima podataka. Primjerice, kako bi se ostvarila potpora Lambda arhitekturi, istovremeno se isti podaci gutaju u jezero podataka i pohranjuju u privremeni spremnik za potporu sloju brzih podataka.

U odnosu na model međusloja predložen u [46], upravitelji u jedinstvenom sloju pristupa podacima preuzimaju ulogu servisa pristupa podacima i upravitelja pohrane podataka. Samo razdvajanje u predloženom modelu posljedica je fokusiranosti isključivo na pohranu podataka te rad sa serijama podataka. Upravitelj pristupa podacima prvenstveno ima zadaću preoblikovanja podataka, što u kontekstu jezera podataka mora biti razloženo na nekoliko

servisa, odnosno upravitelja, koji mogu imati zadatak preoblikovanja podataka. Uzevši u obzir da također obavlja specifične zadatke međusloja, podjela na više servisa koji preuzimaju tu zadaću (poput, primjerice, upravitelja privremenog spremnika neobrađenih podataka).

Zadaće upravitelja pohrane podataka u unificiranom sloju pristupa podacima također su podijeljene na nekoliko servisa. Kako je jedna od zadaća prilagodba podataka u konačni oblik, upravitelj preslikavanja struktura podataka odgovara navedenim zahtjevima. U trenutku poprimanja konačnog oblika prilagođenog određenom modelu pohrane podataka, jedino preostaje podatke usmjeriti na odgovarajuće sučelje. Komunikacija prema samom sloju pohrane podataka obavlja se modelom objava i pretplata, što je zadaća upravitelja pretplata. Iz navedenog je vidljivo kako su zadaće jedne komponente iz predloženog međusloja pristupa podacima u jednakom opsegu pokrivene kroz zaduženja više upravitelja u unificiranom sloju pristupa podacima.

Upravitelji mogu obavljati operacije nad podacima različitih razina kompleksnosti. Operacije sežu od jednostavnog usmjeravanja podataka na temelju prethodno definiranih pretplata, pohrane podataka u međuspremnik do pokretanja kompleksnih operacija integracije i obogaćivanja podataka.

U nastavku su opisani upravitelji koji su uključeni u model jedinstvenog sloja pristupa podacima.

**Stvarnovremenski upravitelj sučelja**

Stvarnovremenski upravitelj sučelja ima zadaću ostvarivanja komunikacije između ulaznih sučelja, na koja pristižu serije i tokovi podataka, i ostalih upravitelja sadržanih u podatkovnim cjevovodima. Pomoću njega se detektira novi skup podataka pristigao na ulazna sučelja. Kako sučelja u sebi ne sadrže mehanizme komunikacije, uloga stvarnovremenskog upravitelja je upućivanje upita za novim podacima prema sučelju odnosno, kroz odgovarajuća sučelja, upite prema izvorima podataka.

Mehanizmi komunikacije oslanjaju se na periodičke upite korištenjem propitkivanja (*eng. poll*) u zadanim vremenskim intervalima. Unutar stvarnovremenskog upravitelja sučelja sadržana su dva načina komunikacije, metodama guranja (eng. *push*) i povlačenja (eng. *pull*).

Metoda guranja upita ostvaruje se komunikacijom između izvora podataka i ulaznog sučelja. Pritom ulazno sučelje po završetku prihvata podataka od strane izvora podataka obavještava upravitelja o novom skupu podataka.

Povlačenje podataka ostvaruje se slanjem upita prema ulaznom sučelju i provjerom je li dostupni skup podataka promijenjen u odnosu na rezultate prethodnog upita. U tom slučaju, izvor podataka osigurava mehanizam dostave podataka prema ulaznom sučelju, a sučelje je zaduženo za privremenu pohranu posljednjih dostavljenih podataka.

Prilikom povlačenja podataka, potrebno je osigurati različite mehanizme potpore za serije i tokove podataka. Naime, u slučajevima serija podataka, na ulaznim sučeljima mogu se očekivati podaci koji su već pohranjeni u perzistentnu memoriju te je nužno osigurati mehanizam izbjegavanja pohrane duplih podataka. S druge strane, kada su u pitanju tokovi podataka, povlačenje podataka se oslanja na prozore s određenim vremenskim okvirom i pojava eventualnih istovjetnih zapisa nastaje kao posljedica definicije prozora. U slučajevima kada se u prozorima očekuje pojava prethodno pohranjenih podataka, nužan je odgovarajući mehanizam praćenja, poput praćenja vremenske oznake posljednjeg pohranjenog zapisa.

U obje metode, stvarnovremenski upravitelj sučelja podatke s ulaznog sučelja prosljeđuje u podatkovni cjevovod i tako započinje proces gutanja podataka.

**Upravitelj preslikavanja tokova podataka u serije podataka**

Upravitelj preslikavanja tokova podataka u serije podataka preuzima ulogu pretvorbe podataka iz prozora toka podataka u seriju podataka. U ovom slučaju, ne dolazi do suštinske promjene nad podacima, već isključivo obogaćivanja podataka procesnim metapodacima i eventualnim izvedenim atributima.

Jedan od primjera uloge ovog upravitelja je eksplicitna dodjela vremenske oznake pojedinom zapisu u slučaju da je on ne nosi sa sobom. U tom slučaju, vremenska oznaka preuzima se iz samog toka podataka i izvodi iz metapodataka prozora u kojem je zapis dostavljen. Također, procesni metapodaci obogaćeni u upravitelju dobivaju se komunikacijom s repozitorijem metapodataka u svrhu primjene procesa upravljanja podacima te ne zadiru u suštinu zapisa. Upravitelj preslikavanja tokova podataka dio je procesa gutanja podataka, no njegovo uključivanje u podatkovne cjevovode nije uvijek nužno.

**Upravitelj preslikavanja struktura podataka**

Zadaća upravitelja preslikavanja struktura podataka je primjena operatora poput *relation-to-document, blob-to-document* i sl. nad zapisima pristiglima kroz ulazna sučelja u podatkovne cjevovode. Na taj način ostvaruju se preslikavanja između strukturiranih i polustrukturiranih te između nestrukturiranih i polustrukturiranih zapisa korištenjem prethodno definiranih operatora.

**Upravitelj spajanja podataka**

Upravitelj spajanja podataka obavlja zadatke integracije strukturiranih te spajanja polustrukturiranih podataka. Sam upravitelj ne sadrži implementacije integracijskih postupaka niti postupaka spajanja, već zadatke obavlja korištenjem odgovarajućih sustava, programskih okvira i programa. Pozivanje operacija integracija i spajanja obavlja se na temelju odgovarajućih zapisa u repozitoriju metapodataka za pojedine entitete.

Unutar upravitelja spajanja podataka, ukoliko nisu implementirani u odgovarajućim sustavima, predviđeno je i obavljanje osnovnih operacija provjere kvalitete i pročišćivanja podataka. Budući da je jedan od preduvjeta uspješnog jezera podataka održavanje razine kvalitete podataka i korištenje procesa upravljanja podacima, neodgovarajuće podatke nužno je identificirati prije no što budu upisani u perzistentnu memoriju. Iz tog razloga, upravitelj spajanja podataka potreban je u svakom izvedenom podatkovnom cjevovodu.

**Upravitelj repozitorija metapodataka**

Zadaća upravitelja repozitorija metapodataka je ostvarivanje dvosmjerne komunikacije između repozitorija metapodataka i upravitelja koji se nalaze u unificiranom sloju pristupa podacima. Na taj način osigurano je centralno mjesto koje na jednak način pruža mogućnost dohvata i pohrane podataka u sam repozitorij te zadržavanje istovjetnosti struktura podataka i podataka o procesima kroz cijelo jezero podataka.

Upravitelj jezera podataka suštinski sakriva kompleksnost repozitorija, prilagođava strukturu metapodataka za komunikaciju s preostalim upraviteljima. Također, pomoću njega omogućuje se uniformno upravljanje procesima oslonjenima na repozitorij metapodataka.

**Upravitelj pretplata**

Izlazna sučelja unificiranog sloja pristupa podacima ne posjeduju mehanizme za prihvat podataka, već se podaci na njih dostavljaju korištenjem mehanizma pretplata. U svrhu upravljanja pretplatama potrebno je uvesti zasebnog upravitelja koji podržava sustav temeljen na paradigmi objavi / pretplati (eng. *publish / subscribe*). Upravitelj pretplata nužan je dio podatkovnog cjevovoda, budući da ujedno kontrolira i ispravnost dostave podataka prema izlaznim sučeljima.

**Upravitelj privremenog spremnika neobrađenih podataka**

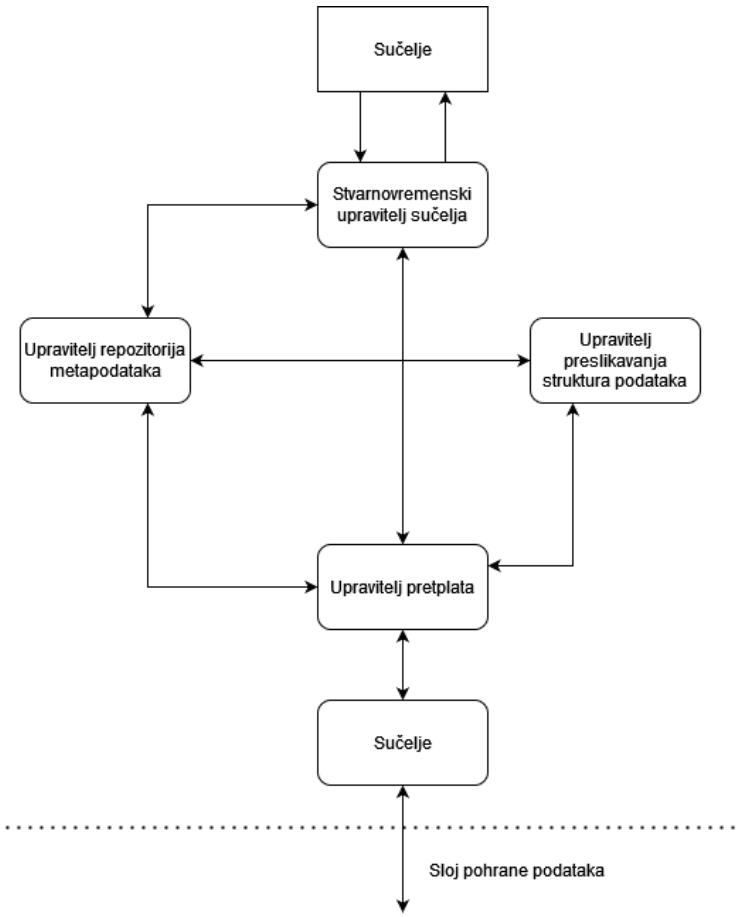
U svrhu potpore Lambda arhitekturi, potrebno je osigurati privremenu pohranu podataka prije njihovog upisa u perzistentnu memoriju. Za to je zadužen upravitelj privremenog spremnika neobrađenih podataka, koji podatke pristigle u podatkovni cjevovod brzog sloja

podataka pohranjuje u spremnik. Time se osigurava njihova dostupnost kroz specijalizirana izlazna sučelja, do trenutka kad njihova obrada završi i budu upisani u perzistentnu memoriju.

Upravitelj se, pored upisivanja podataka, brine i o njihovom pravovremenom micanju iz privremenog spremnika na temelju podataka dobivenih od preostalih upravitelja (npr. upravitelja pretplata koji dojavljuje uspješnu isporuku podataka).

### 3.5.5. Primjeri izvedbe modela i podatkovnih cjevovoda

Na slici 15 prikazan je apstraktni podatkovni cjevovod potpunog gutanja podataka od prihvata na ulaznom sučelju do dostupnosti na izlaznom sučelju prema sloju pohrane podataka.



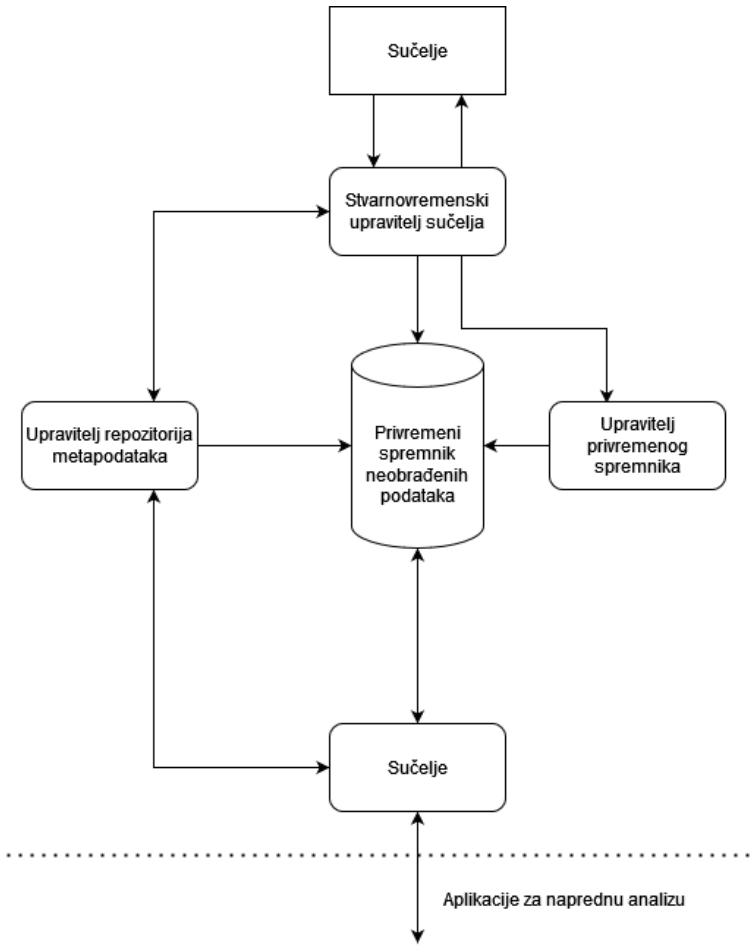

Slika 15. Dijagram cjevovoda potpunog gutanje podataka

Svako sučelje ima vlastitu instancu stvarnovremenskog upravitelja sučelja, koji njime upravlja na temelju metapodataka dobivenih iz repozitorija. Upravitelj je zadužen za postavljanje vrijednosti značajki pojedinog sučelja, poput načina dohvata podataka (*push / pull*), veličine vremenskog okvira ukoliko sučelje prihvaća podatke iz tokova podataka i preostalih značajki definiranih metapodacima.

Podaci iz upravitelja sučelja dostavljaju se upravitelju pretplata. Upravitelj pretplata prihvaća podatke, nakon čega se podaci na temelju zadanih metapodataka mogu proslijediti prema odgovarajućem izlaznom sučelju ili prema upravitelju preslikavanja struktura podataka.

Podatkovni cjevovod sa slike 15 ima ulogu sloja serije podataka u Lambda arhitekturi, budući da se podaci prije pohrane u trajnu memoriju preslikavaju i integriraju, ovisno o definiciji u repozitoriju metapodataka. Time je ostvarena potpora kompleksnijem modelu jezera

podataka koje uključuje odgovarajuća prilagođavanja podataka, a ujedno i mogućnost zadržavanje procesa integracije podataka ukoliko se skladište podataka nalazi unutar jezera podataka.




Slika 16. Dijagram cjevovoda gotovo stvarnovremenskog gutanja podataka

Na slici 16 prikazan je podatkovni cjevovod podataka koji odgovara arhitekturi sloja brzih podataka u Lambda arhitekturi. Njegova uloga je prvenstveno potpora aplikacijama za naprednu analizu koje zahtijevaju gotovo stvarnovremensko gutanje podataka sa što manjim odmakom od trenutka pristizanja. U tom slučaju, potrebno je osigurati podatke iz najnovijih prozora tokova podataka, koji se u nepromijenjenom obliku pohranjuju u privremeni spremnik neobrađenih podataka. Na taj način, podaci koji još nisu upisani u trajnu memoriju, odnosno ne pripadaju sloju serije podataka, postaju dostupni za obrade. Stvarnovremenski upravitelj sučelja zadužen je pritom za upis podataka u privremeni spremnik, gdje su oni pohranjeni u radnoj memoriji. Podaci se iz privremenog spremnika miču temeljem definiranih granica i razlikuju se u pristupima micanju.

Prvi pristup temelji se na metapodacima i prozorima, pri čemu pojedini upravitelj sučelja ima definiran broj prozora podataka koje pohranjuje u privremeni međuspremnik kroz metapodatke. Zadaću brisanja podataka iz najstarijeg pohranjenog prozora, nakon što njihov red istekne, ima upravitelj privremenog spremnika. Pojedini stvarnovremenski upravitelj sučelja obavještava upravitelja privremenog spremnika o dolasku novog prozora i broju dozvoljenih prozora u međuspremniku. Važno je istaknuti da prozori mogu biti i kližući (*eng. sliding*) i kotrljajući (*eng. tumbling*), što je parametar pojedinog sučelja definiran unutar samog repozitorija metapodataka. Neovisno o vrsti prozora, upravitelj nakon isteka granice briše podatke pristigle iz pojedinog sučelja.

Drugi pristup je postavljanje vremenskog ograničenja kroz metapodatke, pri čemu pojedini upravitelj šalje obavijest upravitelju privremenog spremnika o micanju podatka starijih od određene vremenske granice, bez vođenja brige o pripadnosti pojedinom prozoru.

Treći pristup je globalno definirani parametar vremenske granice. U tom pristupu upravitelji sučelja nemaju zadaću oko upravljanja micanja podataka iz privremenog spremnika, a navedena zadaća prepuštena je upravitelju privremenog spremnika neobrađenih podataka.

## 3.6. Model repozitorija metapodataka

### 3.6.1. Uloga repozitorija metapodataka

Repozitorij metapodataka zauzima važno mjesto u arhitekturi jezera podataka. Kvalitetno uspostavljen repozitorij metapodataka omogućava unificiranu obradu podataka, praćenje strukture i evolucije podataka. Uz pohranjenu strukturu, pomoću metapodataka podržano je i uvođenje praćenja podrijetla podataka kroz transformacije zadane nizom operacija između pojedinih entiteta.

Podjela metapodataka ima dva generalna pristupa: podjelu na intraobjektne i interobjektne metapodatke te podjelu na strukturne, svojstvene i semantičke metapodatke. Navedeni pristupi fokusirani su na entitete i njihove članove, no nisu dostatni za praćenje obrada podataka. U tu svrhu uvode se dodatne vrste metapodataka koji se baziraju na prethodno opisanim entitetima, no primarna zadaća im je opis pojedinih obrada podataka.

Repozitoriji metapodataka već se dulje vrijeme koriste kao potpora u procesu skladištenja podataka, iako tamo njihova implementacija nije nužna za ispravan rad. Pojedini alati za integraciju podataka interno grade repozitorije metapodataka koji su uglavnom sakriveni od korisnika, tako da oni nisu često ni svjesni njihovog postojanja. No, za uspješnu i cjelovitu implementaciju procesa vladanja podacima (eng. *data governance*), koncepta upravljanja podacima na mikrorazini pojedinačnog jezera podataka, ključno je integrirati model repozitorija metapodataka u opsegu koji te procese podržava.

Metapodaci su prema okviru za upravljanje podacima DAMA-DMBOK [12] jedan od stupova vladanja podacima i isprepleteni su sa svim segmentima procesa vladanja podacima. Svi segmenti se u većoj ili manjoj mjeri oslanjaju na dostupne metapodatke.

S tehničke strane, repozitorij metapodataka može se koristiti kao temelj dinamičke izgradnje sustava za obradu i pohranu podataka. Sami podaci mogu se opisati kroz proizvoljno definirane hijerarhije, gdje se na najnižoj razini opisuju članovi koji tvore pojedine zapise. Kako se hijerarhija grupira prema višim razinama, obrada podataka ujedno se opisuje i funkcijama koje se vežu za odgovarajućeg člana u toj hijerarhiji.

Predloženi model repozitorija podataka slijedi podjelu na interobjektne i intraobjektne metapodatke i opisan je odgovarajućim klasama. Pored njih, identificirane su i potrebe za

metapodacima za potporu obrada podataka i metapodacima za praćenje evolucije podataka. Navedeni metapodaci također su opisani kroz klase.

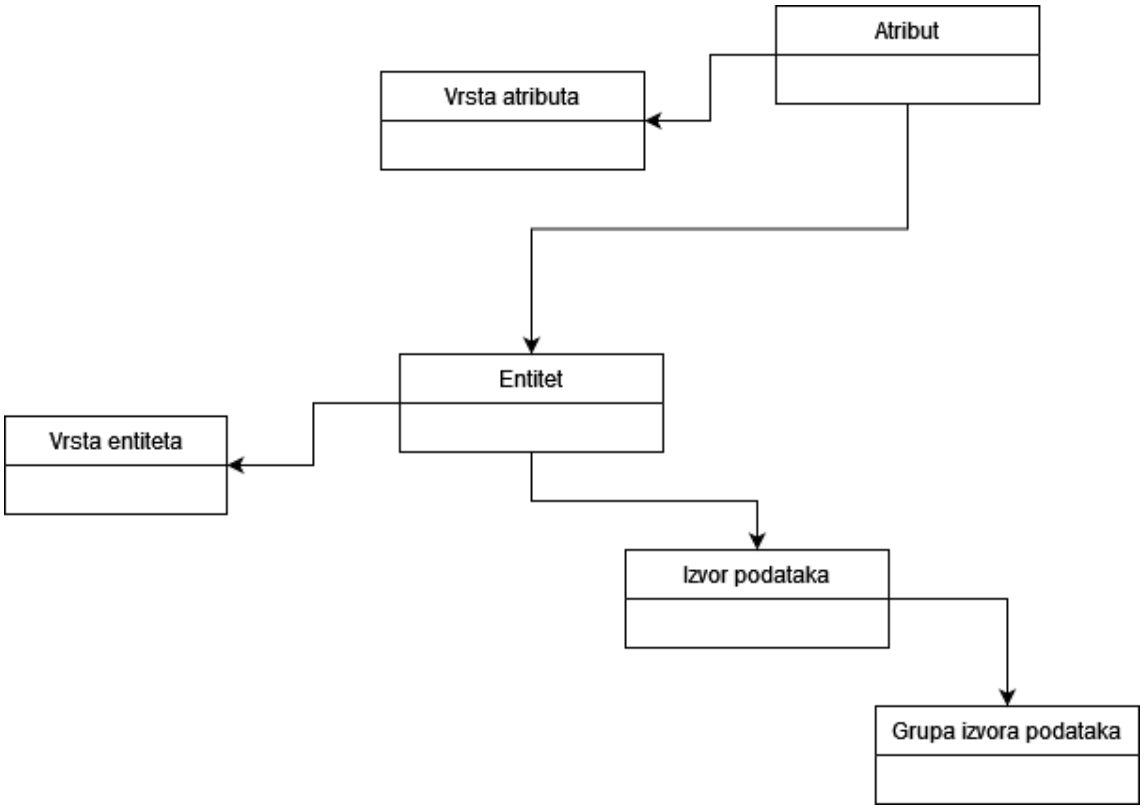
U svrhu potpore procesima upravljanja podacima, interobjektni i intraobjektni metapodaci mogu se proširiti dodatnim značajkama i klasama. Na taj način repozitorij metapodataka postaje centralno mjesto pohrane i opisivanja procesa te omogućuje dinamičko upravljanje podacima, pri čemu se proces prilagođava zapisu na kojeg se odnosi kroz skup prethodno definiranih pravila.

### 3.6.2. Intraobjektne klase metapodataka

Na temelju prethodne definicije iz [8], definicija intraobjektnih klasa metapodataka prilagođena je na sljedeći način:

**Definicija 15.** Intraobjektna klasa metapodataka definira pojedini objekt i uključuje atribute kojima se definiraju sadržaja objekta, struktura objekta, statistički podaci o objektu i informacije i korištenju skupa podataka koji objekt predstavlja.

Na slici 17 prikazan je dijagram odnosa intraobjektnih klasa metapodataka u repozitoriju metapodataka




Slika 17. Intraobjektne klase metapodataka

U hijerarhiji intraobjektnih klasa, na najvišem stupnju apstracije je klasa grupa izvora podataka u kojoj su objedinjeni izvori podataka prema korisnički definiranom skupu zajedničkih značajki. Grupu izvora podataka opisujemo kôdom, nazivom i opis grupe.

Tablica 3. Atributi klase grupa izvora podataka

| Atribut | Vrsta atributa |
|---|---|
| Kôd grupe izvora podataka | string |
| Naziv grupe izvora podataka | string |
| Opis grupe izvora podataka | string |

U klasi izvora podataka opisane su značajke pojedinog izvora podataka. Svaki izvor podataka objedinjuje skup objekata, koji ne moraju nužno biti iste strukture. Značajke kojima opisujemo izvor podataka su njegov jedinstveni identifikator (kôd), naziv izvora podataka i njegov opis, uz oznaku pripadnosti grupi izvora podataka.

Tablica 4. Atributi klase izvor podataka

| Atribut | Vrsta atributa |
|---|---|
| Kôd izvora podataka | string |
| Naziv izvora podataka | string |
| Opis izvora podataka | string |

Klasa entitet prva je od klasa u hijerarhiji intraobjektnih klasa koje detaljnije opisuju podatke, odnosno u njezinom slučaju jedinstveni skup podataka unutar pojedinog izvora podataka. Pored identifikatora entiteta, naziva i opisa, potrebno je entitet opisati značajkama vezanima uz obrade tokova podataka: frekvencijom dolaska novih zapisa, veličinom otvora zapisa (eng. *window*) toka podataka i kvantifikatorom otvora zapisa. Uz klasu su vezane dvije klasifikacije: tip kvantifikatora otvora zapisa i struktura. Klasifikacija tipa kvantifikatora otvora zapisa ukazuje radi li se o otvoru temeljenom na zadanom broju očitanih vrijednosti (diskretni) ili o otvoru ograničenim vremenskim okvirom (vremenski). Klasifikacija strukture označava radi li se o strukturiranim, polustrukturiranim ili nestrukturiranim zapisima.

Klasa vrste entiteta upućuje je li objekt konačni ili neprekinuti skup podataka (tok podataka), radi li se o datoteci u jezeru podataka ili tablici / zapisu i sl.

Tablica 5. Atributi klase entitet

| Atribut | Vrsta atributa |
|---|---|
| Kôd entiteta | string |
| Naziv entiteta | string |
| Opis entiteta | string |
| Struktura | klasifikacija (strukturirani, polustrukturirani, nestrukturirani) |
| Frekvencija dolaska zapisa | time |
| Veličina otvora zapisa | integer |
| Kvantifikator otvora zapisa | klasifikacija (diskretni, vremenski) |

Važno je istaknuti postojanje klasifikacije, koja opisuje je li pojedini entitet strukturirani, polustrukturirani ili nestrukturirani. Svaka od navedenih klasifikacija ukazuje na strukturu, a posljedično i način obrade i pohrane podataka sadržanih u tom entitetu. Iznimno je važno ispravno klasificirati strukturu pojedinog entiteta, budući da se na taj način ispravno usmjerava slijed podataka unutar unificiranog sloja pristupa podacima prema pojedinim sučeljima.

Podatke najdetaljnije opisuje klasa atributa. Opisana je značajkama identifikatora i naziva atributa, klasifikacijom vrste atributa, duljinom atributa u bajtovima, značajkama preciznosti i skale koje se popunjavaju isključivo za numeričke atribute te značajkama vezanim uz kontrolu kvalitete podataka – oznakom obaveznog polja

Tablica 6. Atributi klase atribut

| Atribut | Vrsta atributa |
|---|---|
| Kôd atributa | string |
| Naziv atributa | string |
| Opis atributa | string |
| Tip atributa | string |
| Duljina atributa | int |
| Preciznost atributa | int |
| Skala atributa | int |
| Obavezno polje | boolean |

Svakoj instanci klase entitet pridružena je jedna ili više instanci klase atribut, ovisno o njegovoj strukturi. Tako su nestrukturirani zapisi opisani entitetom koji uz sebe imaju jedan atribut pomoću kojeg je definiran cijeli njegov zapis s maksimalnom očekivanom duljinom te vrstom atributa koja reprezentira način zapisivanja podataka. Polustrukturirani i nestrukturirani zapisi, pak, mogu sadržavati više instanci klase atribut za koje su poznati vrsta zapisa i njihova duljina. U slučaju polustrukturiranih zapisa, skup svih instanci klase atributa ukazuje na sve atribute koje možemo očekivati u navedenom entitetu. Time je omogućena fleksibilna reprezentacija pojedinog polustrukturiranih entiteta, što je jedna od njegovih bitnih značajki. Strukturirani objekti, s najmanjom mogućom potporom za promjene, nužno moraju sadržavati sve instance klasa atributa dostupnih za obradu.

U tablici 7 dan je popis podržanih promjena u intraobjektnim klasama repozitorija metapodataka.

Tablica 7. Popis podržanih promjena u intraobjektnim klasama repozitorija metapodataka

| Grupa izvora podataka | Evolucijsko praćenje | Neaktivno stanje |
|---|---|---|
| Dodanje nove grupe izvora podataka | Da | Ne |
| Uklanjanje postojeće grupe izvora podataka | Da | Da |
| Izvor podataka | | |
| Dodavanje novog izvora podataka | Da | Ne |
| Promjena pripadnosti grupi izvora podataka | Da | Ne |
| Uklanjanje postojećeg izvora podataka | Da | Da |
| Entitet | | |
| Dodavanje novog entiteta | Da | Ne |
| Promjena izvora podataka za entiteta – uključuje atribute i relacije | Da | Ne |
| Preimenovanje entiteta– uključuje atribute i relacije | Da | Ne |
| Promjena vrste strukture entiteta – uključuje atribute i relacije | Da | Ne |
| Uklanjanje postojećeg entiteta | Da | Da |
| Atribut | | |
| Dodavanje novog atributa | Da | Ne |

| | | |
|---|---|---|
| Preimenovanje atributa – uključuje relacije | Da | Ne |
| Promjena vrste atributa – uključuje relacije | Da | Ne |
| Uklanjanje postojećeg atributa | Da | Da |

Ključno je istaknuti da u modelu repozitorija metapodataka nema fizičkog brisanja zapisa, budući da se na taj način gubi mogućnost praćenja evolucije podataka. Nasuprot tome, za označavanje izbrisanog podatka koriste se oznake izbrisano i aktivno koje su tipa *boolean* te pri mijenjaju vrijednosti u *točno* i *netočno*. Dodatna mogućnost je pritom i ponovno aktiviranje pojedinog zapisa.

U nastavku bit će na primjeru polustrukturiranog i strukturiranog zapisa prikazan način njihovog opisivanja unutar repozitorija metapodataka.

**Strukturirani zapis**

Izvor podataka je tok podataka na programskom sučelju s kodom GPS_API_S (grupa izvora podataka API_S), a korišteni entitet je TrackPoint (kod TP_S). Atributi, uz pripadne vrijednosti, prikazane su u tablici 8.

Tablica 8. Primjer strukturiranog zapisa

| Vrijeme | Pozicija | HR_BPM | Brzina | Kadenca |
|---|---|---|---|---|
| 2022-04-27 16:05:06 | Point (45.803568697465824, 15.960449958220124) | 87 | 0.0560000017285347 | 0 |

U repozitoriju metapodataka, ovaj zapis bit će prikazan na sljedeći način (opisi su izostavljeni radi čitljivosti).

**Izvor podataka:**

| Kôd izvora podataka | Naziv izvora podataka | Opis izvora podataka |
|---|---|---|
| GPS_API_S | Strukturirani GPS API | ... |

**Entitet** (radi preglednosti struktura zapisa je prikazana u recima) iz izvora podataka GPS_API_S**:**

| | |
|---|---|
| Kôd entiteta | TP_S |
| Naziv entiteta | TrackPoint |
| Opis entiteta | ... |
| Struktura | Strukturirani |
| Frekvencija dolaska zapisa | 00:00:01 |
| Veličina otvora zapisa | 1 |
| Kvantifikator otvora zapisa | vremenski |

**Atributi**

| Kôd atributa | Naziv | Opis | Tip | Duljina | Preciznost | Skala | Obavezno polje |
|---|---|---|---|---|---|---|---|
| 1 | Vrijeme | ... | Time | 8 | 8 | 0 | T |
| 2 | Pozicija | ... | Point | 22 | | | T |
| 3 | HR_BPM | ... | Int | 4 | | | N |
| 4 | Brzina | ... | Float | 4 | | | N |
| 5 | Kadenca | ... | Int | 4 | | | N |

**Polustrukturirani zapis**

Izvor podataka je tok podataka na programskom sučelju s kodom GPS_API_P (grupa izvora podataka API_P), a korišteni entitet se također naziva TrackPoint (kod TP_P). Polustrukturirani zapis prikazan je kao LPAV.

```
TrackPoint: {
   "Vrijeme":"2022-04-27 16:05:06",
   "Pozicija": {
        "type":"Point",
        "coordinates": [45.803568697465824, 15.960449958220124]
        },
   "HR_BPM":87,
   "Brzina":0.0560000017285347,
   "Kadenca":0
}
```

Iz strukture podataka vidljivo je kako je atribut Pozicija GeoJSON zapis prostornog tipa *point*, te se na takav način i zapisuje u metapodatke.

U repozitoriju metapodataka, ovaj zapis bit će prikazan na sljedeći način (opisi su izostavljeni radi čitljivosti).

**Izvor podataka:**

| Kôd izvora podataka | Naziv izvora podataka | Opis izvora podataka |
|---|---|---|
| GPS_API_P | Polustrukturirani GPS API | ... |

**Entitet** (radi preglednosti zapis je pivotiran)**:**

| | |
|---|---|
| Kôd entiteta | TP_P |
| Naziv entiteta | TrackPoint |
| Opis entiteta | ... |
| Struktura | Polustrukturirani |
| Frekvencija dolaska zapisa | 00:00:01 |
| Veličina otvora zapisa | 1 |
| Kvantifikator otvora zapisa | vremenski |

**Atributi**

| Kôd atributa | Naziv | Opis | Tip | Duljina | Preciznost | Skala | Obavezno polje |
|---|---|---|---|---|---|---|---|
| 6 | Vrijeme | ... | Time | 8 | 8 | 0 | T |
| 7 | Pozicija | ... | Point | 22 | | | T |

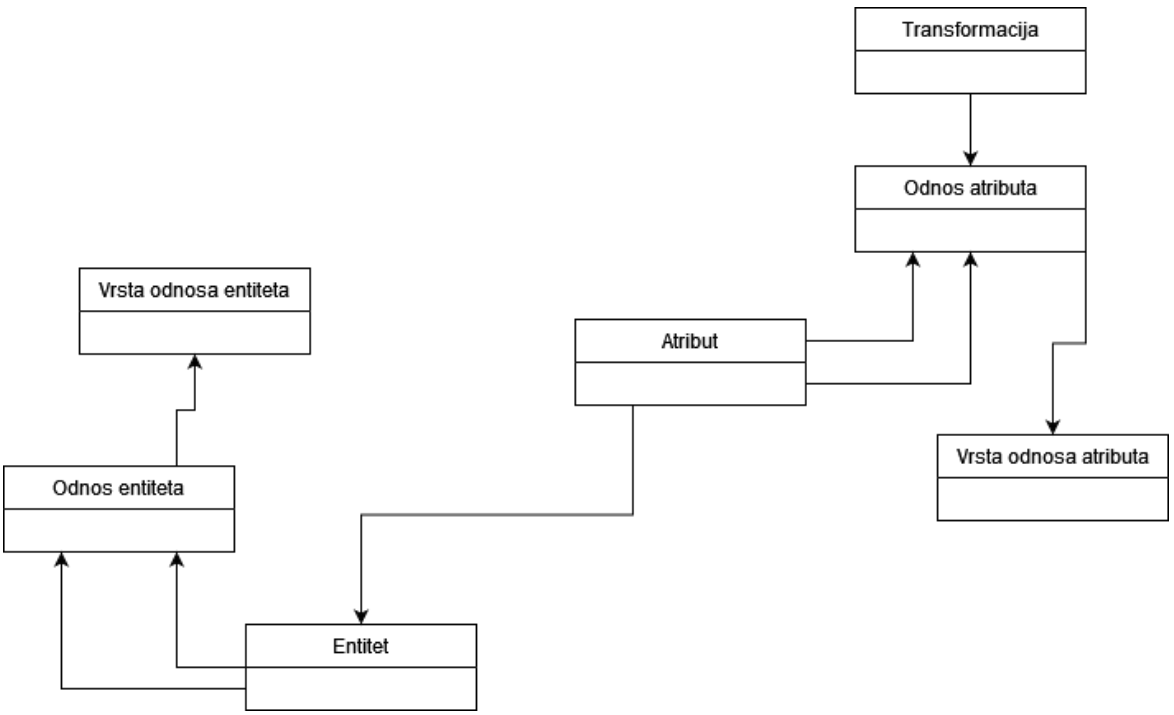
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 8 | HR_BPM | ... | Int | 4 | | | N |
| 9 | Brzina | ... | Float | 4 | | | N |
| 10 | Kadenca | ... | Int | 4 | | | N |

### 3.6.3. Interobjektne klase atributa

Na temelju prethodne definicije iz [8], definicija interobjektnih klasa metapodataka prilagođena je na sljedeći način:

**Definicija 16.** Interobjektne klase podataka specificiraju odnose između različitih skupova podataka ili između atributa koji pripadaju različitim skupovima podataka.

Interobjektne klase metapodataka sadrže ograničenja podataka i svojstva koji opisuju spojnosti različitih skupova podataka i razinu srodnosti između različitih atributa [8]. Dijagram interobjektnih klasa podataka prikazan je na slici 18.




Slika 18. Interobjektne klase metapodataka, zajedno s pripadnim vezanim intraobjektnim klasama

Tablica 9. Popis podržanih promjena u interobjektnim klasama repozitorija metapodataka

| Transformacija | Evolucijsko praćenje | Neaktivno stanje |
|---|---|---|
| Dodavanje transformacije | Da | Ne |
| Uklanjanje transformacije | Da | Da |
| Odnos | | |
| Dodavanje odnosa | Da | Ne |
| Uklanjanje odnosa | Da | Da |

Klase odnosa (za entitete i atribute) imaju sličnu strukturu, koja je u tablici 10 prikazana za odnos entiteta, a u tablici 11 za odnos atributa. U terminologiji unutar klasa odnosa koriste se termini „izvorni“ (eng. *source*) i „odredišni“ (eng. *target*), budući da su odnosi usmjereni.

Tablica 10. Atributi odnosa entiteta

| Atribut | Vrsta atributa |
|---|---|
| Kôd izvornog entiteta | string |
| Kôd odredišnog entiteta | string |
| Kôd vrste odnosa entiteta | string |

Tablica 11. Atributi odnosa atributa

| Atribut | Vrsta atributa |
|---|---|
| Kôd izvornog atributa | string |
| Kôd odredišnog atributa | string |
| Kôd vrste odnosa atributa | string |

Koristeći primjere iz poglavlja 3.6.3, kao primjer odnosa bit će prikazana potpora za transformacije korištenjem metapodataka.

Prvenstveno, odnosi između dva entiteta u ovom primjeru u repozitoriju metapodataka iskazani su na sljedeći način (kôd vrste odnosa entiteta kao primjer ima vrijednost IO, što je kratica za izvor-odredište):

| Kôd izvornog entiteta | Kôd odredišnog entiteta | Kôd vrste odnosa entiteta |
|---|---|---|
| TP_N | TP_P | IO |

Kod vrste odnosa entiteta prikazan je kao IO, no njegova vrijednost ovisi o zapisima koji pripadaju klasi vrsta odnosa entiteta. Analogno, odnosi između atributa iskazani su u metapodacima kao:

| Kôd izvornog atributa | Kôd odredišnog atributa | Kôd vrste odnosa atributa |
|---|---|---|
| 1 | 6 | T |
| 2 | 7 | T |
| 3 | 8 | T |
| 4 | 9 | T |
| 5 | 10 | T |

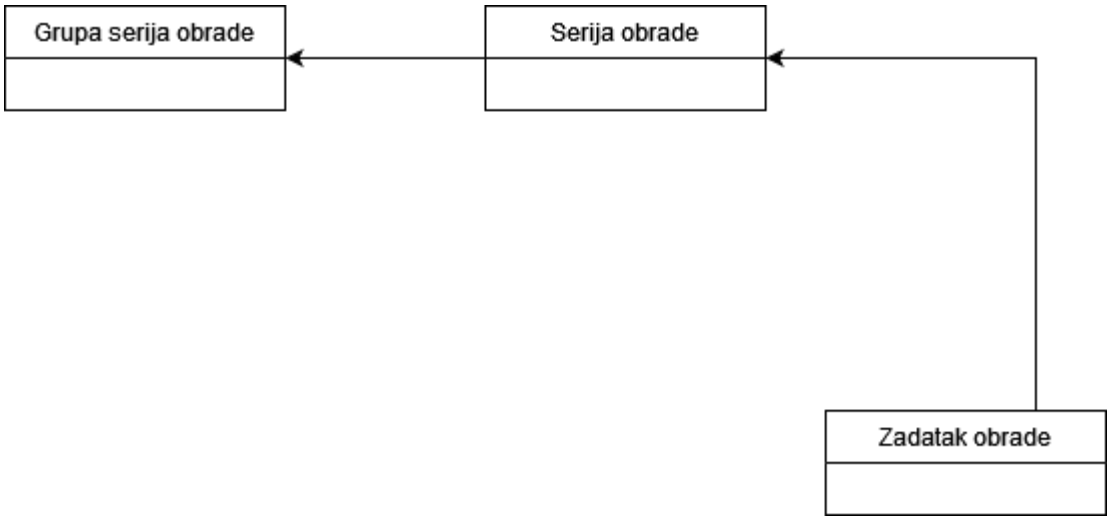
Budući da između dva entiteta i njihovih atributa u primjeru nema transformacija tipa podataka niti spajanja više atributa u jedan, nisu prikazani dodatni metapodaci transformacija. Primjer transformacija iskazanih metapodacima bila bi, primjerice, promjena tipa podataka atributa Brzina iz *float* u *numeric* preciznosti 28 i skale 10 decimala.

### 3.6.4. Klase obrade podataka

U klase obrade podataka spadaju sve klase u kojima se ne opisuju podaci, već grupe obrada podataka. Svrha uvođenja klasa obrada podataka jest omogućavanje praćenja obrada serija podataka, no nisu isključivo ograničene na njih. Podaci iz tokova podataka također mogu imati koristi od dodatne razine praćenja koja nije ograničena isključivo na vremensku oznaku, stoga su klase obrade podataka modelirane na način da podržavaju unificirano obrade serija i tokova podataka.

Klase obrada podataka obuhvaćaju klase skupa grupa obrada podataka, grupa obrada podataka, zadataka obrada podataka, hijerarhije grupa obrada podataka i hijerarhije zadataka obrada podataka.

Na slici 19 prikazan je dijagram klasa obrade podataka




Slika 19. Dijagram klasa obrade podataka

Klasa skupa grupa obrada podataka je hijerarhijski najviša i predstavlja logički povezane grupe obrada podataka. Unutar pojedinog skupa grupa obrada podataka definiraju se ovisnosti između grupa obrada podataka kroz značajke sljednosti i ograničenja uspješnog završetka izvršavanja pojedine grupe. Značajka sljednosti označava redoslijed izvršavanja grupa obrada podataka unutar jednog skupa, pri čemu kao nasljednik izvršavanja jedne grupe može biti više međusobno neovisnih grupa. Kroz ograničenje uspješnog završetka dajemo skupu grupa informaciju utječe li neuspješno završena grupa obrada podataka negativno na nastavak obrade, odnosno hoće li greška zaustaviti daljnju obradu povezanog niza grupa obrada podataka.

Tablica 12. Popis podržanih promjena u klasama obrade podataka repozitorija metapodataka

| Skup grupa obrade | Evolucijsko praćenje | Neaktivno stanje |
|---|---|---|
| Dodavanje skupa grupe obrade | Da | Ne |
| Uklanjanje skupa grupe obrade | Da | Da |
| Grupa obrade | | |
| Dodavanje grupa obrade | Da | Ne |
| Uklanjanje grupa obrade – uključuje i zadatke obrade | Da | Da |
| Dodavanje grupa obrade u skup grupa serija obrade | Da | Ne |
| Uklanjanje grupa obrade iz skupa grupa serija obrade | Da | Ne |
| Zadaci obrade | | |

| | | |
|---|---|---|
| Dodavanje zadatka obrade | Da | Ne |
| Uklanjanje zadatka obrade | Da | Da |
| Dodavanje hijerarhije zadataka obrade | Da | Ne |
| Uklanjanje hijerarhije zadataka obrade | Da | Ne |

Iako klase obrade podataka predstavljaju podatke koji se ne mijenjaju toliko često, nužno je pratiti njihovu evoluciju. Promjene procesa koje su dovele do određenih obrada podataka često se reflektiraju na njihov konačni oblik i sadržaj. Prilikom praćenja evolucije i podrijetla podataka, svaki proces unutar pojedinog podatkovnog cjevovoda utječe na konačni rezultat.

Iz tog razloga, pored intraobjektnih i interobjektnih klasa podataka, klase evolucije podataka prate promjene i u podacima klasa obrade podataka. Opis načina praćenja evolucije u ovim klasama bit će dan u nastavku, prilikom opisa klasa evolucije podataka.

### 3.6.5. Klase evolucije podataka

Klase evolucije podataka podržavaju praćenje promjene definicija metapodataka kroz vrijeme. Potreba za praćenjem promjena definicija proizlazi iz nužnosti za potporom obrade i pohrane polustrukturiranih i nestrukturiranih podataka, pri čemu je važno poznavanje korištene ispravne strukture u danom vremenskom trenutku.

Kroz klase evolucije podataka prate se promjene u interobjektnim i intraobjektnim metapodacima, što uključuje definicije grupe izvora podataka, izvora podataka, entiteta, atributa, vrsta odnosa i transformacija. Praćenje promjena analogno je praćenju izmjena u sporo promjenjivim dimenzijama tipa 7 u skladištima podataka, uz dodatak klasifikacije vrste promjene. Na taj način podaci koji pristižu u obrade iz tokova podataka prate se prema trenutnoj strukturi zapisa, dok se obrade serija podataka prate prema strukturi iz odgovarajućeg trenutka. Unutar svake od klasa prati se verzija kao neprekinuti cjelobrojni niz na način da se promjenom definicije ujedno povećava verzija.

Dok dodavanje i promjena novog zapisa povećavaju vrijednost verzije i rezultiraju novim zapisom u odgovarajućoj klasi, isključivanje zapisa iz pojedine strukture provodi se na način da se posljednji ispravan zapis zatvara s vrijednošću vremenske oznake kraja valjanosti. Time se ostavlja mogućnost ponovne aktivacije zapisa u budućnosti jednostavnom operacijom uz zadržavanje postojeće sljednosti i povijesti promjena.

## 3.7. Algoritmi unutar UDAL

U kontekstu obrade podataka postoji veliki broj algoritama koji pokrivaju različita područja obrade podataka, od jednostavnih algoritama za rad nad skupovima podataka do kompleksnih algoritama poput spajanja entiteta.

Unutar UDAL ugrađeni su algoritmi za rad s tokovima podataka. Navedeni algoritmi opisani su na apstraktnoj razini i prilagođeni su predloženom modelu unificiranog sloja pristupa podacima. Navedeni algoritmi temelj su kompleksnijih podatkovnih cjevovoda koji imaju zadaću integracije strukturno heterogenih podataka u jedinstven tok podataka i pohranu u perzistentnu memoriju.

Algoritme unutar unificiranog sloja pristupa podatcima dijelimo u dvije glavne skupine. U prvoj skupini nalaze se apstraktni algoritmi spajanja tokova podataka, dok se u drugoj nalaze algoritmi potpore upraviteljima unutar unificiranog sloja pristupa podatcima. Pored algoritama navedenih u nastavku poglavlja postoje algoritmi specifični za prostorne, odnosno prostorno-vremenske podatke, no kako nisu predmet istraživanja, neće biti posebno obrađeni.

### 3.7.1. Algoritmi vezani za spajanje tokova podataka

Tokovi podataka u unificirani sloj pristupa podacima ulaze kroz niz međusobno odvojenih sučelja. Dio tokova podataka pritom dijeli skup atributa, bilo da se radi o LPAV (opisanim u poglavlju 2.3) ili strukturiranim ntorkama. U svrhu kreiranja novih LPAV koji te tokove podataka objedinjuju, kada sa strane analize podataka ima smisla, potrebno je obaviti operacije spajanja tokova podataka. Jedna od funkcionalnosti unutar unificiranog sloja pristupa podacima je i integracija podataka u zajedničku strukturu, koja se potom pohranjuje u perzistentnu memoriju, odnosno odgovarajuću zonu u jezeru podataka.

Prilikom integracije tokova podataka važna su dva algoritma spajanja tokova podataka: spajanje oblika LPAV – LPAV i spajanje oblika LPAV – relacija. Rezultat operacija spajanja je skup LPAV i kao takav je dostupan u podatkovnim cjevovodima, odnosno posljedično u perzistentnoj memoriji. Algoritmi se teoretski mogu koristiti neograničeno, budući da nisu ograničeni vrstom prozora, već je fokus na samoj logici kako se heterogeni polustrukturirani tokovi podataka spajaju u jedan tok sa zajedničkom shemom.

U algoritmima se koristi sljedeća terminologija, preuzeta iz poglavlja 3.1.4:

Neka je $\mathcal{S}$ tok podataka nad shemom $S$. Neka je $A$ podskup atributa sheme $S$.

Algoritam spajanja LPAV – LPAV koristi operatore definirane u poglavlju 2.3.1., koji će biti ujedno i dio samog algoritma. Nužno je naglasiti kako je algoritam definiran na apstraktnoj razini s dva toka podataka, dok algoritam prilikom implementacije može koristiti neograničen skup tokova podataka kao ulazne parametre.

**Algoritam 1** Spajanje tokova podataka oblika LPAV i LPAV

```
procedure ST_SJOIN(S1, S2)
    define S_JOINED = Object_Merge(S1, S2)
    create new list of elements list_elements with schema S_JOINED
    for each s in (S1, S2)
        create new element e with schema S_JOINED
        for each a in A(s)
            if a exists in A(S_JOINED)
            add value from a(s) to a(e)
            else
            set value NULL for a(e)
        add element e into list_elements
        end for each
    end for each
    return list_elements
end procedure
```

Prije algoritma spajanja tokova podataka oblika LPAV i ntorka, potrebno je definirati način stvaranja sheme LPAV iz sheme relacije. U poglavlju 3.1.4 definiran je operator stvaranja sheme iz relacije, kojem nadalje u apstraktnim algoritmima predstavlja funkcija Get_Shema_From_Relation ($R$). Funkcija za zadanu ntorku $R$ vraća pripadnu shemu $S$.

Algoritam spajanja LPAV i ntorke logikom je istovjetan algoritmu spajanja tokova podataka LPAV i LPAV, uz razliku da se kao drugi parametar u funkciji očekuje relacijski tok podataka. Pretpostavka algoritma jest da toku podataka $\mathcal{S}_2$ pripada relacijska shema $R_2$. Rezultat algoritma je lista LPAV, pri čemu su u apstraktnom algoritmu definirana dva toka podataka, dok ih u implementaciji može biti teoretski neograničen broj.

**Algoritam 2** Spajanje tokova podataka oblika LPAV i ntorka

```
procedure ST_SRJOIN(S1, S2)
    define S2 = Get_Shema_From_Relation (R2)
    define S_JOINED = Object_Merge(S1, S2)
    create new list of elements list_elements with schema S_JOINED
    for each s in (S1, S2)
        create new element e with schema S_JOINED
        for each a in A(s)
            if a exists in A(S_JOINED)
            add value from a(s) to a(e)
            else
            set value NULL for a(e)
        add element e into list_elements
        end for each
    end for each
    return list_elements
end procedure
```

### 3.7.2. Algoritmi za potporu upraviteljima UDAL

Sljedeći skup algoritama odnosi se na potporu upraviteljima unificiranog sloja pristupa podacima. Prvi dio algoritama u ovom skupu odnosi se na preslikavanja samih zapisa između heterogenih struktura (ntorka u LPAV i nestrukturirani zapis u LPAV), dok je drugi dio algoritama vezan za specifičnosti pojedinih upravitelja.

Unutar jezera podataka, prilikom pohrane podataka u zone prilagođenih podataka i zone prilagođenih pristupa, potrebno je podatke pohranjene u zoni nepromijenjenih podataka strukturni prilagoditi u LPAV. Za preslikavanje između struktura, prvenstveno iz ostalih struktura u LPAV, zadužen je posebni upravitelj koji se uključuje u podatkovne cjevovode.

Algoritmi 3 i 4 prikazuju način preslikavanja ntorke, odnosno nestrukturiranog zapisa tipa *blob*, u LPAV. Ova su preslikavanja vrlo jednoznačna, pogotovo pri preslikavanju nestrukturiranih zapisa, iz razloga što ne postoje hijerarhijski vezani podaci.

Sam algoritam preslikavanja ntorke u LPAV temelji se na ranije definiranoj operaciji dohvaćanja sheme iz relacije, opisane u poglavlju 3.1.4. Pretpostavka algoritma jest da se u

relaciji i shemi istovjetni atributi nalaze na istim pozicijama. Nakon stvaranja sheme i pripadne instance sheme kao objekta, vrijednosti koje pripadaju atributima u ntorki preslikavaju se u vrijednosti istovjetnih zapisa u shemi.

**Algoritam 3** Preslikavanje zapisa ntorke u LPAV

```
procedure TUPLE_TO_SCHEMA (R_1)
    define S_1 = Get_Shema_From_Relation (R_1)
    create new object s with schema S_1
    for each id in (R_1)
        set value v_i of a_i to have value v_i of id_i
    end for each
    return s
end procedure
```

Algoritam preslikavanja nestrukturiranog zapisa tipa *blob* u LPAV iznimno je jednostavan – kako nestrukturirani zapis sadrži samo jednu vrijednost, ona se pridodaje atributu unutar LPAV.

**Algoritam 4** Preslikavanje zapisa *blob* u LPAV

```
procedure BLOB_TO_SCHEMA (B_1)
    define S_1 = {(a_1, v_1)}
    create new object s with schema S_1
    set value v_1 of a_1 to have value v_1 of B_1
    return s
end procedure
```

Kako je navedeno u poglavlju 3.5.4, u unificiranom sloju pristupa podacima nalazi se stvarnovremenski upravitelj sučelja sa zadaćom osvježavanja podataka na dolaznim sučeljima i njihovim slanjem u povezane podatkovne cjevovode. Sam upravitelj u sebi sadrži implementaciju algoritma 5, koji je u svojoj apstraktnoj formi prikazan niže. Algoritam se temelji na podacima o sučeljima iz repozitorija metapodataka (interval osvježavanja, vrsta dohvata podataka u sučeljima) i provjeri novih podataka u definiranim intervalima.

**Algoritam 5** Algoritam osvježavanja stvarnovremenskog upravitelja sučelja

```
procedure MANAGER_TICK_REFRESH (R1)
    get refresh interval t from metadata repository for interfaces
    get fetching method definitions for interfaces
    find latest data refresh timestamp for each interface
    for each i in interfaces
        if current timestamp is greater than refresh timestamp + t
            case when i is pull type
                fetch new window with data
                if window is not empty, send data to data pipeline
            when i is push type
                check interface for new data
                send data to data pipeline
        end if
    end for each
end procedure
```

Konačno, algoritam 6 povezan je s implementacijom Lambda arhitekture unutar unificiranog sloja pristupa podacima. Sam algoritam je, kao i prethodni, apstrahiran te se može prilagoditi arhitekturi izvedbe unificiranog sloja pristupa podacima.

**Algoritam 6** Algoritam upravljanja zapisima u privremenom spremniku

```
procedure REFRESH_CACHE (R1)
    for each i in interfaces
        make t_ii as latest timestamp of fetched data for i
        make t_io as latest timestamp of persistently stored data for i
        delete all data older than t_io from cached data
        union data from interface i with cached data
    end for each
end procedure
```

# 4. Pregled alata za naprednu analizu

U pregledu alata za naprednu analizu obavljena je kvalitativna analiza alata prisutnih na tržištu koji trenutno spadaju među korištene ili značajnije unutar svoje domene. Alati su podijeljeni u nekoliko kategorija, ovisno o značajkama te su za svaku od kategorija ispitane mogućnost obrade prostornih i prostorno vremenskih analiza.

## 4.1. Alati opće namjene

Prva kategorija obuhvaća alate za naprednu analizu opće namjene. Na tržištu je dostupan značajan broj generičkih alata s ugrađenim funkcijama za implementaciju dubinske analize podataka, strojnog učenja, vizualizacija, obrade podataka. Glavna im je odlika orijentiranost prema krajnjem korisniku, kojem pružaju mogućnost implementacije vlastitih rješenja kroz vizualna sučelja uz mogućnost pisanja vlastitog koda.

Četiri su primjera takvih alata obuhvaćena u ovoj analizi: Alterix, DataIku, Tibco i SAS. Prema Gartnerovom istraživanju platformi za podatkovnu znanost i strojno učenje iz 2021 [52], ovi alati spadaju u sam vrh platformi, stoga su se nametnuli kao vrijedni subjekti analize.

U kontekstu prostornih i prostorno-vremenskih podataka, svi navedeni alati nude potporu za prostorne podatke, bez eksplicitne potpore za prostorne podatke u vremenskoj domeni. No, kako se radi o alatima koji omogućuju proširivost pisanjem vlastitog koda, korisnici su u mogućnosti proširiti postojeće naredbe za rad s prostornim tipovima podataka u prostorno-vremenske.

Tablica 13 daje osnovni pregled alata za naprednu analizu opće namjene u kontekstu prostorno-vremenskih podataka.

Tablica 13. Usporedba alata za naprednu analizu opće namjene

| Alat | Prostorni tipovi podataka | Primjeri podržanih operacija | Prostorno-vremenska potpora |
|---|---|---|---|
| Alterix [53] | DA (WKT, WGS84) | Najkraća udaljenost, udaljenost, najbliži objekt | NE eksplicitno |
| DataIku [54] | DA (WKT, WGS84) | GeoJoin (Geo-NN), udaljenost, geoBuffer, geoWithin, geoContains | NE eksplicitno |

| | | | |
|---|---|---|---|
| SAS [55] | DA (SAS/GIS Data Model) | Coverage skup operacija (Contains, prostorno filtriranje) | NE eksplicitno |
| Tibco [56] | DA (WKB) | GreatCircleDistance (najkraća udaljenost između dvije točke) | NE eksplicitno |

Alterix i DataIku podržavaju WKT prostorne tipove podataka (point, line, polygon...) koji su sukladni skupu podržanih tipova podataka predloženih u poglavlju 3.1.2. Također, podržani su predloženi operatori, a posjeduju i implementaciju korisnih algoritama.

SAS koristi SAS/GIS model, koji tek djelomično podržava skup podržanih tipova podataka. U SAS/GIS modelu podržani su tipovi podataka analogni WKT podacima iz predloženog skupa podržanih tipova podataka.

Tibco, za razliku od ostalih alata, ne podržava WKT zapise prostornih tipova podataka već isključivo podržava WKB. Interno, zapisi se preslikavaju u oblik *Geometry* koji je specifičan za Tibco ekosustav alata. Osnovne operacije nad prostornim tipovima podataka su podržane unutar alata.

## 4.2. Proširenja programskih okvira

U drugu kategoriju analiziranih alata spadaju proširenja programskih okvira. U kontekstu rada s prostorno-vremenskim podacima analizirat će se proširenja programskih okvira Spark i Flink, koji spadaju u često korištene okvire. Analiza ne obuhvaća rad samih programskih okvira i njihove mogućnosti, već je fokus isključivo na dostupnim proširenjima.

Unutar programskog okvira Spark, uspoređena su proširenja STARK [57], GeoSpark [58] i SpatialSpark [59]. Budući da su sva proširenja temeljena na prostornim podacima, ujedno su njima i pokriveni definirani prostorni tipovi podatka iz poglavlja 3.1.2, a sam programski okvir Spark podržava tip podatka *blob*. Ujedno, sva uspoređena proširenja podržavaju rad s programskim jezicima Scala i Java.

Za razliku od proširenja GeoSpark i SpatialSpark, STARK podržava rad s prostorno-vremenskim tipovima podataka i vremensko podizanje prostornih operacija. Također, STARK ima prednost nad ostalim proširenjima u vidu prevodioca Piglet koji omogućava izradu skripti s mogućnosti izvršavanja ne samo u programskom okviru Spark, već i u programskom okviru Flink.

Za programski okvir Flink analizirano je proširenje GeoFlink. GeoFlink podržava rad s prostorno-vremenskim tokovima podataka, u potpunosti podržava skup tipova podataka i operacija definiranih u poglavlju 3.1.2. Poput prethodno navedenih proširenja za programski okvir Spark, podržava rad s programskim jezicima Scala i Java.

Tablica 14. Usporedba proširenja programskih okvira Spark i Flink

| | Programski okvir | Podržani programski jezici | Potpora za prostorno-vremenske tipove podataka |
|---|---|---|---|
| GeoSpark | Spark | Scala, Java | Ne |
| SpatialSpark | Spark | Scala, Java | Ne |
| STARK | Spark / Flink | Scala, Java | Da |
| GeoFlink | Flink | Scala, Java | Da |

Posebno mjesto u analizi potrebno je posvetiti projektu Apache Sedona [60], nastalom iz projekta GeoSpark. Iako je u njemu izostala potpora za prostorno-vremenske tipove podataka i operacije nužne za potporu prostorno-vremenskim tokovima podataka, nudi određene prednosti. Prvenstveno se one očituju u obliku potpore većem broju programskih jezika (Python, R i Spatial SQL) kroz aplikacijska programska sučelja. Na taj način se potpora za prostorno-vremenske tipove podatka i operacije može ostvariti dodatnim proširenjima u obliku korisnički definiranih funkcija.

## 4.3. Proširenja programskih jezika bibliotekama

U fokusu analize proširenja programskih jezika, uzevši u obzir odnos volumena strukturiranih, polustrukturiranih i nestrukturiranih podataka, analizirat će se programski jezici Python, R i Scala. Upitni jezik SQL nije dio analize, budući da su prostorni tipovi podataka u njemu prethodno definirani kao dio standarda SQL-99. Proširenja upitnog jezika CQL prikazana su u radu [32] i također nisu dio ove analize.

Programski jezik Python široko je korišten u alatima za naprednu analizu. Njegova je upotreba rezultat više faktora (jednostavnost pisanja koda, proširivost i brzo izvođenje su primjeri). Kao posljedica rasprostranjenosti korištenja razvijene su nove biblioteke, pomoću kojih su osnovne funkcije programskog jezika proširene novim tipovima podataka i operacijama.

U svrhu rada s prostornim i prostorno vremenskim podacima koriste se programske biblioteke GeoPandas, sptemp, MovingPandas, PySal i PyMove. U sljedećoj je tablici dana matrica potpore prostornim, odnosno prostorno-vremenskim tipovima podataka.

Tablica 15. Biblioteke programskog jezika Python

| | Prostorni tipovi podataka | Prostorno-vremenski tipovi podataka |
|---|---|---|
| GeoPandas [62] | Da (WKT) | Ne |
| sptemp [63] | Da (WKT) | Da (WKT) |
| MovingPandas [64] | Da (WKT) | Da (trajektorije) |
| PySal [65] | Da (WKT) | Da |
| PyMove [66] | Da (WKT) | Da |

GeoPandas je programska biblioteka nastala iz projekta otvorenog koda koja proširuje programsku biblioteku *pandas* operacijama nad prostornim podacima. MovingPandas, pored prostornih podataka, proširuje biblioteku GeoPandas potporom za operacije nad trajektorijama. PyMove je biblioteka slična MovingPandas sa sintaksom biblioteke *pandas* koja omogućava prostorno-vremenske operacije nad podacima.

Programska biblioteka sptemp tipovima podataka i operacijama u potpunosti pokriva skup tipova podataka naveden u poglavlju 3.1.2, uz mogućnost proširenja dodatnim korisničkim funkcijama. Biblioteka PySal sadrži module za rad s prostornim i prostorno-vremenskim tipovima podataka i povezanim operacijama, kao i naprednu analizu tih podataka.

Programski jezik R ponajviše je zastupljen u statistici. Za razliku od programskog jezika Python, proširenja su sadržana u paketima, umjesto u bibliotekama. Osnovni paket s prostorno-vremenskim tipovima podataka i operacijama je *spacetime*, na temelju kojeg su nastali brojni paketi koji koriste njegove klase. Napredna analiza prostorno-vremenskih podataka u programskom jeziku R omogućena je kroz veliki broj dediciranih paketa, poput SpatioTemporal [67], STMedianPolish [68] i sličnih.

Programski jezik Scala može se proširiti bibliotekom GDAL [69], koja pruža potporu za rad s prostorno-vremenskim podacima koji odgovaraju skupu tipova podataka definiranih u poglavlju 3.1.2. Značaj programskog jezika Scala u kontekstu prostorno-vremenskih podataka vezan je uz programske okvire Python i Spark i njihova proširenja, navedena u poglavlju 4.2.

## 4.4. Proširenja unutar ekosustava velikih podataka

U ekosustavu velikih podataka značajno mjesto zauzima GeoMesa [70], koja predstavlja skup alata za analizu prostornih i prostorno-vremenskih podataka. Podržan je skup tipova podataka definiranih u poglavlju 3.1.2. te direktno podržava rad s programskim jezicima Java i Scala.

Značaj GeoMesa skupa alata je veliki broj sustava za pohranu podataka nad kojima GeoMesa može obavljati prihvat i pohranu (uz prostorno-vremensko indeksiranje) podataka te povezane analize. Interno, za privremenu pohranu podataka koristi se baza podataka Accumulo temeljena na modelu podataka ključ-vrijednost, dok su ostale baze i sustavi pohrana podataka podržani putem aplikacijskih programskih sučelja. GeoMesa dosljedno prati standarde, stoga su navedena sučelja definirana korištenjem OGC standarda. Uz to, podržani su standardi polustrukturiranih tipova podataka, kao što su GeoJson i GeoAvro. Nad pohranjenim heterogenim podacima moguće je i ostvarivanje virtualizacijskog sloja kroz stvaranje pogleda.

Dodatna prednost GeoMesa skupa alata je mogućnost potpore prostorno-vremenskim tokovima podataka izgradnjom prostorne semantike nad platformom Apache Kafka.

Konačno, GeoMesa uključuje proširenje za rad s programskim okvirom Apache Spark, koji uključuje proširenja za upitni jezik SparkSQL i PySpark programsku biblioteku, čime se efektivno povećava broj podržanih programskih jezika. Zahvaljujući navedenom proširenju, GeoMesa može se koristiti i u oblaku kroz brzorastuću platformu Databricks, prisutnu kod svih vodećih pružatelja cloud usluga.

Unutar Hadoop ekosustava, sustavi za rad sa strukturiranim podacima, Hive i Impala, eksplicitno ne podržavaju rad s prostornim, odnosno prostorno-vremenskim tipovima podataka i operacijama. Stoga je u navedene svrhe nužno proširiti ih korištenjem prikladnih sustava. Uz to, dostupna su proširenja za HDFS datotečni sustav, odnosno programski okvir MapReduce.

Tablični prikaz usporedbe proširenja Hadoop ekosustava za potporu radu s prostornim i prostorno-vremenskim tipovima podataka dan je u tablici 16.

Tablica 16. Usporedba proširenja Hadoop ekosustava za potporu radu s prostorno-vremenskim tipovima podataka

| | Prostorni tipovi podataka | Prostorno-vremenski tipovi podataka | Proširenje Hadoop sustava |
|---|---|---|---|
| SpatialHadoop [48] | Da | Ne | HDFS / MapReduce |

| | | | |
|---|---|---|---|
| ST-Hadoop [49] | Da | Da | HDFS / MapReduce |
| ESRI Hive Spatial | Da | Ne | Hive |
| Sphinx [71] | Da | Ne | Impala |

SpatialHadoop [48] i ST-Hadoop [49] sadrže proširenja za Hadoop ekosustav kojima se na razini samog sustava dodaju proširenja za prostorne i prostorno-vremenske tipove podataka i operacije. Navedena proširenja ugrađena su u vidu semantičke dopune upitnih jezika, dodavanje prostornih i prostorno-vremenskih indeksa nad datotečnim sustavom HDFS i komponenti za programski okvir MapReduce.

Sustav Apache Hive omogućuje rad sa strukturiranim podacima unutar Apache Hadoop ekosustava, pohranjenima u datotečnom sustavu HDFS. U svrhu potpore prostornim podacima ne postoje eksplicitna rješenja, već je sustav moguće proširiti Esri Geometry aplikacijskim programskim sučeljem, odnosno njegovom komponentom Hive Spatial [72]. Pomoću nje ugrađuju se korisnički definirane funkcije u sustav Apache Hive, temeljene na OGC standardu te funkcija za serijalizaciju i deserijalizaciju JSON zapisa iz ArcGIS sustava u strukturirani zapis.

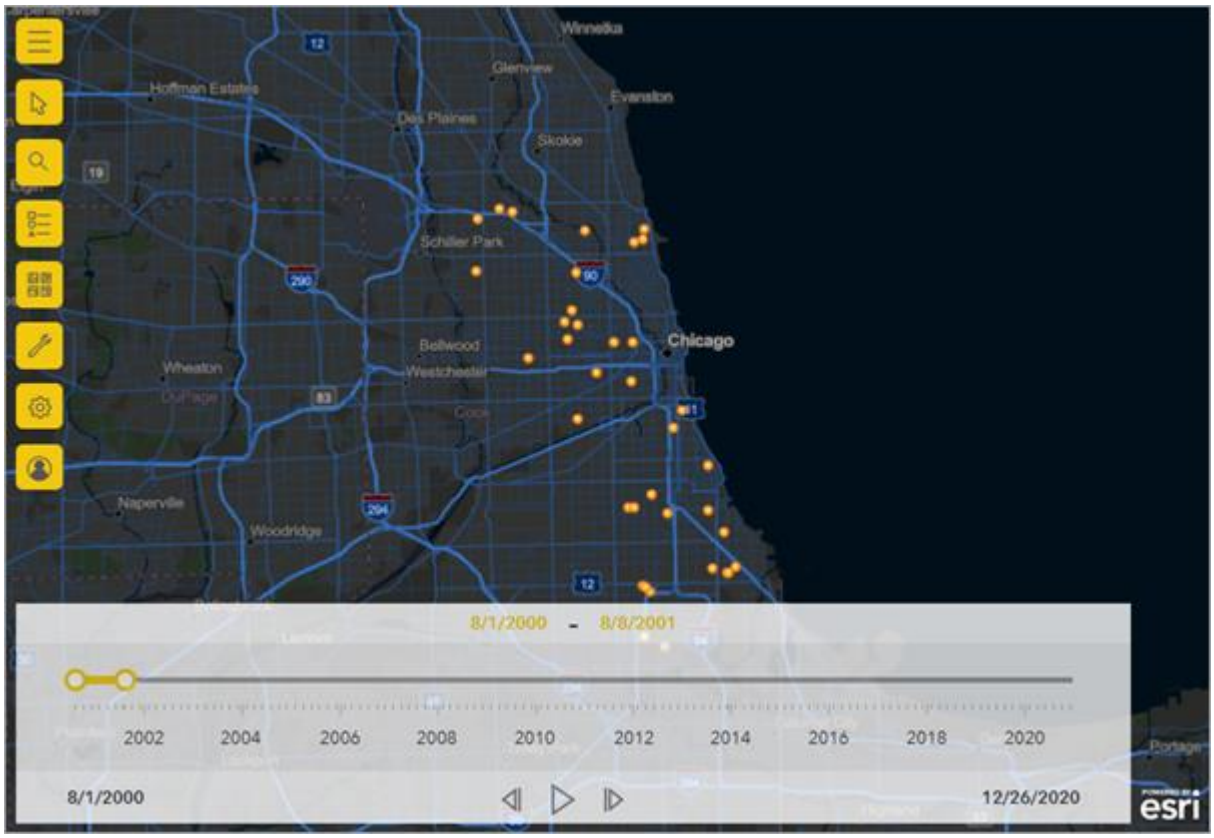
Sphinx [71] je proširenje MPP baze podataka Impala, koja je integrirana s dijelovima Hadoop ekosustava, primjerice sa sustavom Hive kroz dijeljenje repozitorija metapodataka i upitnog jezika. Nastao je na temelju SpatialHadoop, uz prilagodbe za samu bazu Apache Impala. Kao specifičnosti baze, valja istaknuti kako su podaci pohranjeni u nju nepromjenjivi nakon upisa te ne podržavaju transakcije, stoga je ne možemo promatrati kao tradicionalni RDBMS. Pomoću sustav Sphinx, Impala je proširena prostornim podacima i podržanim prostornim operacijama te prostornim indeksima koji se primjenjuju nad podacima. Budući da Apache Impala ne koristi MapReduce programski okvir, proširenja koja se oslanjaju na njega nisu primjenjiva.

## 4.5. Vizualizacijski alati u kontekstu prostorno-vremenskih podataka

Iako vizualizacijski alati nisu eksplicitno u domeni napredne analize, neizostavni su dio svakog analitičkog sustava. Stoga ih je vrijedno kvalitativno analizirati u sklopu potpore radu s prostorno-vremenskim podacima. U analizi nisu obuhvaćena vizualizacijska rješenja iz sustava kao što su GeoMesa ili Python biblioteke, već je fokus isključivo bio na standardnim širokodostupnim vizualizacijskim alatima.

Kvalitativnom analizom obuhvaćena su četiri vizualizacijska alata: Power BI, Qlik, Tableau i Grafana. Bitno je naglasiti da Grafana, iako nije primarno korisnički vizualizacijski alat, u kontekstu prostorno-vremenskih tokova podataka ima svoju svrhu, što će biti objašnjeno kasnije. Preostala tri alata dugogodišnji su lideri u segmentu analitičkih platformi i platformi za poslovnu inteligenciju i podržavaju, uz predefinirane, i korisnički stvarane izvještaje.

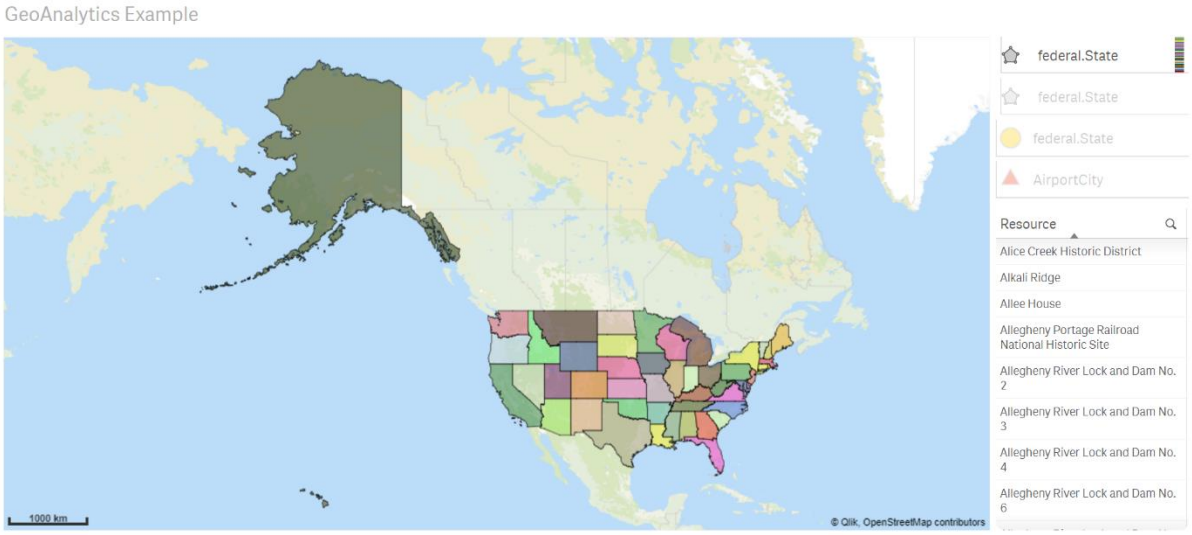
Power BI podržava prostorne podatke putem vizualizacija kroz mape. Prilikom pripreme podataka, identificiraju se prostorne komponente zapisa i pretvaraju u oblik sukladan tipu podataka Point. Uz eksplicitnu potporu za navedene tipove podataka, Power BI alat podržava proširenja putem komponenti za proširenje. Pritom je najznačajnije proširenje za ArcGIS alate, pomoću kojih je skup podržanih tipova podataka proširen za preostale prostorne tipove podataka te je ujedno podržana i vremenska domena u analizi prostorno-vremenskih tipova podataka.




Slika 20. Primjer korištenja PowerBI modela u ArcGIS alatu [73]

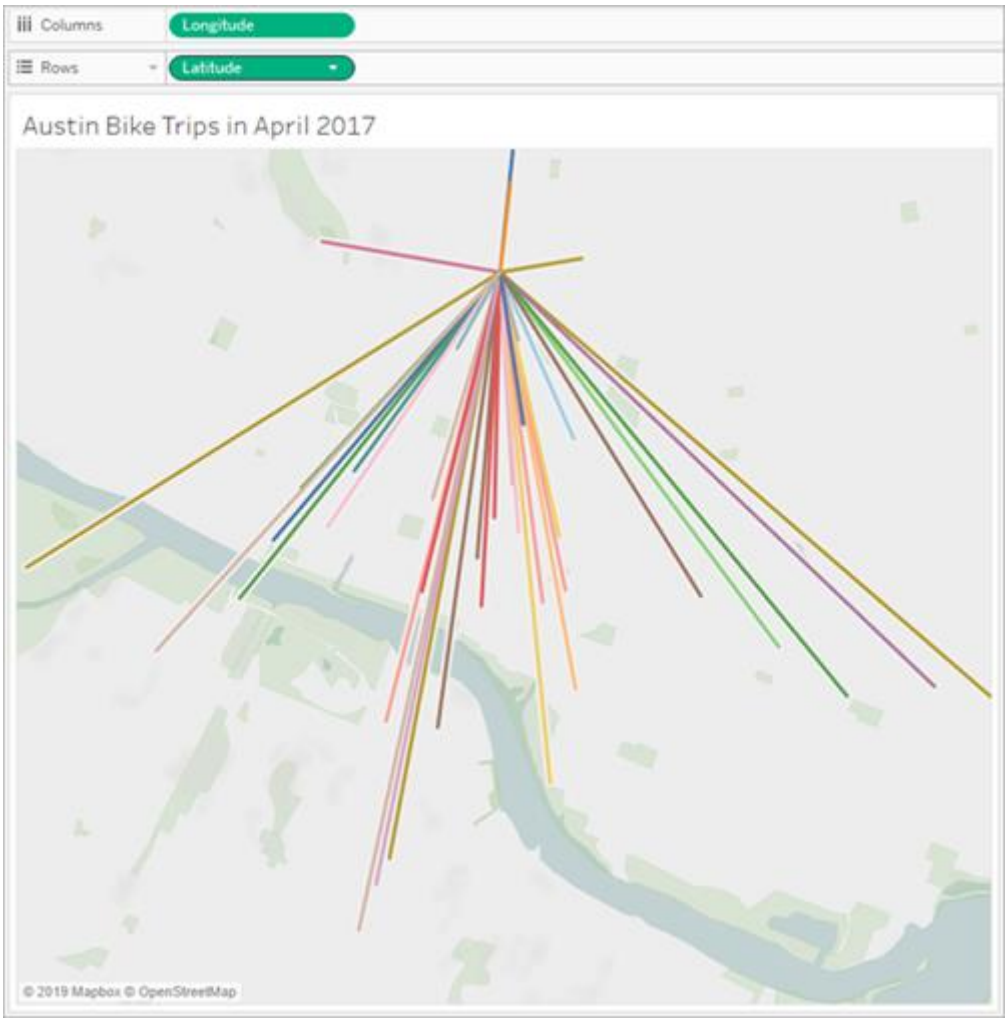
Drugi analizirani alat je Qlik, koji prostorne i prostorno-vremenske podatke podržava kroz skup proširenja GeoAnalytics. Pomoću njega omogućen je rad sa skupom tipova podataka definiranih u poglavlju 3.1.2, uz dodatno proširenje za GeoJSON i ESRI geoprostorne

vektorske (eng. *shapefile*) strukture zapisa. Unutar skupa proširenja GeoAnalytics, definirane su operacije nad tipovima podataka kao dio proširenja GeoOperations.



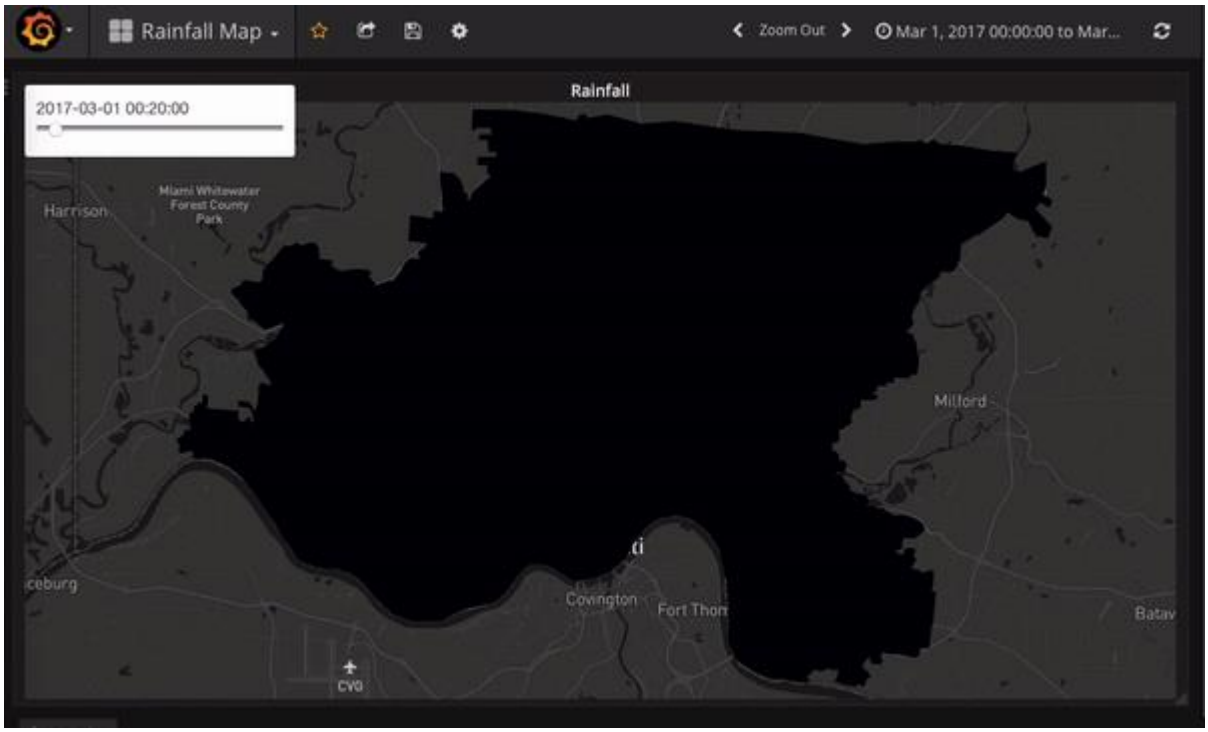

Slika 21. Vizualizacija prostornih podatka u alatu Qlik [74]

Konačno, treći korisnički orijentiran alat je Tableau. U njemu su podržani prostorni tipovi podataka navedeni u poglavlju 3.1.2 kao tip podatka Geometry, a koji se ujedno mogu stvoriti preslikavanjem podataka iz prostornih koordinata u WGS84 zapisu. Od dodatnih struktura zapisa, podržava GeoJSON, ESRI geoprostorne vektorske zapise i TopoJSON. Analiza u vremenskoj domeni moguća je kroz izradu modela unutar alata koji uz prostorne podatke sadrži i njihovu vremensku domenu.




Slika 22. Prikaz prostornih podataka u alatu Tableau [75]

Uz korisnički orijentirane vizualizacijske alate, uzevši u obzir kako jezero podataka s pohranjenim prostorno-vremenskim podacima može imati i svrhu nadzora nad izvorima podataka. Grafana je inicijalno zamišljena kao potpora nadzorom platformama, često naslonjena na baze podataka vremenskih serija. No, uz proširenja, poput proširenja GeoLoop [76], moguće je vizualizirati prostorne podatke, zapisane u GeoJSON formatu, u vremenskoj domeni, kao što je prikazano na slici Slika 23. Prikaz prostorno-vremenskih tokova podataka u alatu Grafana [76].




Slika 23. Prikaz prostorno-vremenskih tokova podataka u alatu Grafana [76]

# 5. Prototip i verifikacija modela jezera podataka

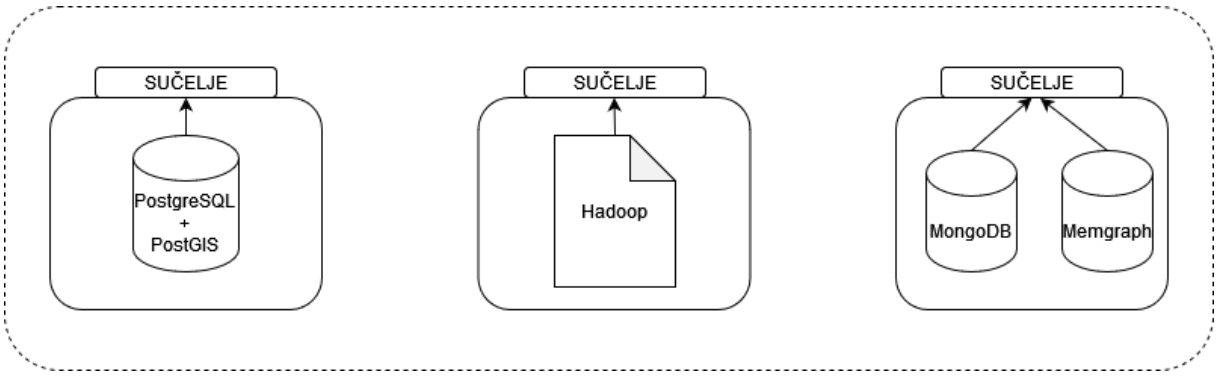
Prototip modela jezera podataka ostvaren je korištenjem kontejnerske (eng. *container*) platforme Docker. Prednost korištenja kontejnera je interoperabilnost između različitih sustava i neovisnost o infrastrukturi. Kontejneri sadrže aplikacije i pripadne biblioteke nužne za rad same aplikacije.

Prototip jezera podataka uključuje više kontejnera, pomoću kojih su realizirani slojevi obrade i pohrane podataka jezera podataka. Nekoliko je bitnih razloga za korištenje kontejnera u odnosu na instalaciju unutar operacijskog sustava:

- ranije navedena neovisnost o infrastrukturi (nema potrebe za dodatnim konfiguracijama u operacijskom sustavu)
- mogućnost višestrukog istovremenog pokretanja istog kontejnera s različitim parametrima
- jednostavno dodavanje novih sustava za obradu ili pohranu podataka kroz kontejnere
- dodavanje kontejnera putem naredbi, čime se može ostvariti automatizacija dijela procesa

## 5.1. Implementacija sloja pohrane podataka

Na slici 24 prikazana je implementacija sloja pohrane podataka u prototipu jezera podataka. Sloj pohrane podataka sadrži kontejnere zadužene za pohranu strukturiranih, polustrukturiranih i nestrukturiranih podataka.




Slika 24. Implementacija sloja pohrane podataka

U svrhu pohrane nestrukturiranih podataka odabran je programski okvir Hadoop. Programski okvir, odnosno njegova distribucija Cloudera, dostupan je kao kontejner, što značajno olakšava samu implementaciju. Unutar programskog okvira za pohranu podataka

koriste se isključivo komponente nužne za pohranu podataka u datotečni sustav HDFS, bez uključivanja dodatnih aplikacija.

Za strukturirane podatke odabrana je postrelacijska baza PostgreSQL, uz pripadno proširenje za rad s prostorno-vremenskim podacima PostGIS. Budući da PostGIS proširenje posjeduje vlastiti kontejner koji ujedno uključuje i PostgreSQL, za rad s PostgreSQL i PostGIS proširenjem dostatno je uključivanje PostGIS kontejnera.

Pohrana strukturiranih podataka nužna je za potporu zonskoj arhitekturi jezera podataka, budući da se u postrelacijsku bazu pohranjuju podaci od zone nepromijenjenih podataka do zone aplikativnog pristupa.

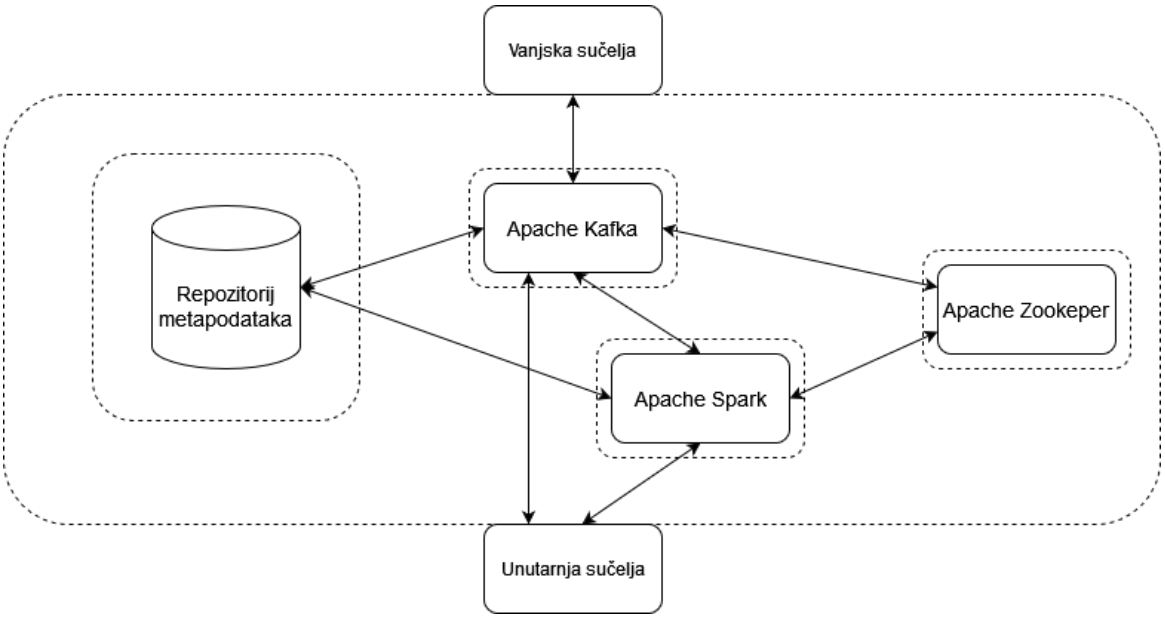
Konačno, potpora pohrani polustrukturiranih podataka ostvarena je kroz kontejnere dokument baze podataka MongoDB i graf baze podataka Memgraph. MongoDB se u proteklom desetljeću nametnuo kao prvi izbor za dokument baze podataka, čime je ujedno omogućava veća razina integracije s programskim jezicima i alatima za analizu podataka. Uz to, u kontekstu prostornih podataka sadrži potporu za prostorne tipove podataka i povezane operacije kroz potporu za GeoJSON strukturu zapisa.

Uz sve navedene sustave za pohranu podataka, ovakav model stvaranja jezera podataka omogućuje jednostavno proširivanje drugim sustavima za pohranu podataka koji se pokažu potrebnima za pojedine vrste analiza. Primjerice, jedan od takvih sustava može biti baza InfluxDB, koja služi za analizu vremenskih serija i putem proširenja Geo podržava rad s prostornim podacima. Uvođenjem takvog sustava, zajedno s pripadnim sučeljem, u zonama istraživanja i aplikativnih pristupa dobiva se potpora za dodatne vrste analiza koje su u postojećim sustavima teško ostvarive ili nepostojeće.

## 5.2. Implementacija sloja obrade podataka

Kao i u slučaju sloja pohrane podataka, osnova implementacije su kontejneri. No, za razliku od sloja pohrane podataka, u sloju obrade podataka nalaze se kontejneri s aplikacijama i sučeljima.

Implementacijski model sloja obrade podataka prikazan je na slici 25.




Slika 25. Implementacija sloja obrade podataka

Kako je definirano u poglavlju 3.3, sloj obrade podataka sadrži repozitorij metapodataka. U implementiranom prototipu jezera podataka, repozitorij je realiziran kao postrelacijska baza unutar kontejnera sa SQL Serverom. Ovdje je potrebno istaknuti kako je opravdano korištenje dvije odvojene postrelacijske baze s različitim ulogama, budući da repozitorij metapodataka treba osigurati potporu svojevrsne automatizacije, poput zakazanog izvođenja nizova procesa, koju PostgreSQL ne omogućava.

Implementacija sučelja putem kontejnera reflektira odliku skalabilnosti jezera podataka i modularnosti, pri čemu je ujedno ostvarena i potpora kontinuiranoj integraciji dorada nad sučeljima. Budući da kontejneri nisu ograničeni operacijskim sustavom, ne postoji niti ograničenje odabira programskog jezika kojim su sučelja izvedena.

Kontejneri s aplikacijama Apache Kafka, Apache Zookeper i Apache Spark, zajedno s ostalim aplikacijama upraviteljima, tvore implementaciju UDAL. Korištenjem aplikacije Apache Kafka realizirana je komunikacija UDAL i sučelja, uključujući unutarnja i vanjska. Uz njih, na razini cijelog sloja obrade podataka aktivni su upravitelji zaduženi za koordinaciju zadataka između aplikacija. Budući da je UDAL u implementacijskom smislu skup aplikacija, važno je odabrati aplikacije i sustave koji se mogu izvoditi u raspodijeljenim sustavima uz zadržavanje funkcionalnosti skalabilnosti i otpornosti na greške.

Unutar sloja obrade podataka, odnosno prilikom implementacije UDAL, ugrađena je potpora radu s prostorno-vremenskim tokovima podataka. Ona je ostvarena prvenstveno kroz implementaciju potpore prostorno-vremenskim tipovima podataka u svim korištenim

aplikacijama, bilo kroz njihova proširenja ili pomoću korisnički definiranih funkcija. Uz njih, ugrađeni su algoritmi opisani u poglavlju 3.7.

## 5.3. Verifikacija modela

Verifikacija modela izvedena je na način da su korištenjem UDAL unutar sloja obrade podataka stvoreni sljedeći podatkovni cjevovodi:

- Stvarnovremenski podatkovni cjevovod sa svrhom pohrane nepromijenjenih podataka u jezero podataka
- Podatkovni cjevovod sa svrhom transformacije podataka iz nepromijenjenog zapisa u zapis prilagođen preostalim zonama

Uz podatkovne cjevovode, stvoren je logički izvještajni model kako bi se pohranjeni podaci vizualizirali kroz aplikaciju izvan jezera podataka. Kako je jedna od zadaća UDAL omogućavanje jedinstvenog pristupa podacima u aplikacijama izvan jezera podataka, ispravnim prikazom podataka iz jezera podataka verificiraju se istovremeno UDAL sučelja i dostupnost podataka u zoni aplikativnog pristupa.

Unutar UDAL potrebno je verificirati potporu za prostorno-vremenske tipove podataka i operacije nad njima unutar pojedinih implementiranih aplikacija na ulaznim i izlaznim sučeljima. Uz njih, verificiraju se ugrađeni algoritmi, neovisno o tome jesu li sastavni dio pojedine aplikacije ili njezino proširenje unutar UDAL.

Izvori podataka su aplikacijska programska sučelja koja sadrže GPS podatke o trenutnoj lokaciji (prostorna točka i visina), vremensku oznaku zapisa, brzinu, kadencu i izmjereni srčani ritam. Podaci se dostavljaju na sučelje kao prostorno-vremenski tok podataka, a u kojem su prikazani isključivo novi podaci. Dinamika dostavljanja novih podataka je jedna sekunda, a sučelje komunikaciju s ostalim sučeljima ostvaruje metodom guranja. U tu svrhu nužno je osigurati neprekidni prihvat podataka na ulaznom sučelju.

Prilikom verifikacije korištena su tri izvora podataka različite podatkovne strukture (strukturirani, nestrukturirani i polustrukturirani), ali u konačnici s istim skupom atributa.

### 5.3.1. Metapodaci

U nastavku su prikazani sažeti metapodaci o korištenim izvorima, entitetima, atributima te transformacijama između pojedinih entiteta, a koje se obavljaju unutar podatkovnih cjevovoda. Radi preglednosti, podaci o izvorima podataka prikazani su u dvije odvojene tablice 17 i 18.

Dio klasa metapodataka proširen je prilikom implementacije u odnosu na predloženi model novim atributima koji nisu dio osnovnog modela repozitorija metapodataka, već su prilikom

implementacije dodani za potporu izgradnje pojedinih cjevovoda (primjer tablice 22) i sučelja (primjer tablica 17 i 18). Također, nisu prikazani svi atributi pojedinih klasa metapodataka

Tablica 17. Metapodaci izvora podataka, dio 1

| Kôd izvora podataka | Struktura | Način dostave |
|---|---|---|
| GPS_API_S | strukturirani | Push |
| GPS_API_P | polustrukturirani | Push |
| GPS_API_N | nestrukturirani | Push |

Tablica 18. Metapodaci izvora podataka, dio 2

| Kôd izvora podataka | Kafka_Topic | API_Path |
|---|---|---|
| GPS_API_S | GPS_S | \localhost\api\gpi_s |
| GPS_API_P | GPS_P | \localhost\api\gpi_p |
| GPS_API_N | GPS_N | \localhost\api\gpi_n |

Osnovni metapodaci entiteta prikazani su u tablici 19, pri čemu je kôd izvora podataka dodan radi prikaza pripadnosti entiteta određenom izvoru podataka.

Tablica 19. Metapodaci entiteta

| Kôd entiteta | Struktura | Frekvencija dolaska zapisa | Kôd izvora podataka |
|---|---|---|---|
| GPS_API_S | strukturirani | 00:00:01 | GPS_API_S |
| GPS_API_P | polustrukturirani | 00:00:01 | GPS_API_P |
| GPS_API_N | nestrukturirani | 00:00:01 | GPS_API_N |

Metapodaci pripadnih atributa prikazani su u tablici 20. Vrijednost kôda entiteta, kao i u prethodnoj tablici, dodan je isključivo u svrhu prikaza pripadnosti atributa određenom entitetu.

Tablica 20. Metapodaci atributa

| Kôd atributa | Naziv atributa | Kôd entiteta | Tip podataka |
|---|---|---|---|
| 1 | Time | GPS_API_S | timestamp |
| 2 | Position | GPS_API_S | point |
| 3 | Altitude | GPS_API_S | float |
| 4 | Distance | GPS_API_S | float |
| 5 | HR_BPM | GPS_API_S | int |
| 6 | Speed | GPS_API_S | float |

| 7 | Cadence | GPS_API_S | int |
|---|---|---|---|
| 8 | Time | GPS_API_P | timestamp |
| 9 | Position | GPS_API_P | point |
| 10 | Altitude | GPS_API_P | float |
| 11 | Distance | GPS_API_P | float |
| 12 | HR_BPM | GPS_API_P | int |
| 13 | Speed | GPS_API_P | float |
| 14 | Cadence | GPS_API_P | int |
| 15 | gps_data | GPS_API_N | blob |

### 5.3.2. Gotovo stvarnovremenski podatkovni cjevovod

Stvarnovremenski podatkovni cjevovod naslanja se na repozitorij metapodataka u kojem su definirani parametri izvora podataka, pripadnih upravitelja i izlaznog sučelja putem kojeg se podaci pohranjuju u perzistentnu memoriju.

Upravitelj stvarnovremenskih sučelja unutar UDAL izveden je kao aplikacija u .NET programskom okviru korištenjem biblioteka za rad i komunikaciju s aplikacijom Kafka. Upravitelj na temelju metapodataka izvora prikuplja podatke s vanjskih aplikacijskih programskih sučelja, prosljeđuje ih prema konfiguriranoj Kafka temi (eng. *topic*) koja ih šalje na sučelja. Radi se o jednostavnoj aplikaciji koja se sastoji od procedure s neprekidnom petljom primanja i slanja poruka prema Apache Kafka te funkcije za slanje poruka. Pojednostavljeni oblik programske potpore (procedura) dan je u nastavku:

```
async procedura Main(kafkaKonfiguracija, tema, intervalOsvježavanja, APIputanja)
{
    U neprekinutoj petlji
    {
          Dohvati asinkrono poruku s API putanje i pohrani u varijablu
          Asinkrono pozovi proceduru SendMessage(kafkaKonfiguracija, tema,
poruka);
          pričekaj intervalOsvježavanja milisekundi
    }
}

async procedura SendMessage(kafkaKonfiguracija, tema, poruka)
{
    Otvori novi KafkaProducer s kafkaKonfiguracijom
          Asinkrono pozovi KafkaProducer(tema, poruka)
    Zatvori KafkaProducer
}
```

Shema arhitekture stvarnovremenskog podatkovnog cjevovoda stvorenog u svrhu verifikacije unificiranog sloja pristupa podacima prikazana je na slici 26.


Slika 26. Arhitektura implementiranog gotovo stvarnovremenskog podatkovnog cjevovoda

Implementirani podatkovni cjevovod ima zadaću gutanja nestrukturiranih podataka, stoga je i potreban manji broj komponenti unutar cjevovoda. U verificiranom modelu, podaci dolaze kao zapis tipa *blob* iz izvora podataka i pohranjuju se u perzistentnu memoriju. Ujedno, nestrukturirani podaci poput prikazanih u verificirajućem modelu jezera podataka mijenjaju strukturu kako bi ih se moglo koristiti u vizualizacijskih alatima.

Na slici 27 prikazan je primjer kontrolnog ispisa dnevničkog zapisa iz stvarnovremenskog upravitelja sučelja:

```
Received 'IlRpbWUiOiAiMjAyMi0wNC0yN1QxNjowNToxMi4wMDBaIiwKICAgICAgICAgICAgICAgICAgICAgIlBvc2l0aW9uIjogewogICAgICAgICAgICAgIC
AgICAgICAgICJMYXRpdHVkZURlZ3JlZXMiOiA0NS44MDM1NzMyMjQ2OTM1NCwKICAgICAgICAgICAgICAgICAgICAgICAiTG9uZ2l0dWRlRGVncmVlcyI6IDE1Ljk2MD
UzOTIyNTQ4ODkwMQogICAgICAgICAgICAgICAgICAgICAgICB9LAogICAgICAgICAgICAgICAgICAgICAgICAiQWx0aXR1ZGVNZXRlcnMiOiAxMjAsCiAgICAgICAgICAgICAgIC
AgICJEaXN0YW5jZU1ldGVycyI6IDcuMTE5OTk5ODg1NTU5MDgyLAogICAgICAgICAgICAgICAgICAgICAgICAiSGVhcnRSYXRlQnBtIjogewogICAgICAgICAgICAgIC
AgICAgICAgICJWYWx1ZSI6IDk0CiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIC
AgICJUUFgiOiB7CiAgICAgICAgICAgICAgICAgICAgICAgICAiU3BlZWQiOiAxLjUwMTk5OTk3NDI1MDc5MzUsCiAgICAgICAgICAgICAgICAgICAgICAgICAiUnVuQ2
FkZW5jZSI6IDg3CiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIC
02:45

Delivered 'IlRpbWUiOiAiMjAyMi0wNC0yN1QxNjowNToxMi4wMDBaIiwKICAgICAgICAgICAgICAgICAgICAgIlBvc2l0aW9uIjogewogICAgICAgICAgICAgICAgI
CAgICAgICAgICJMYXRpdHVkZURlZ3JlZXMiOiA0NS44MDM1NzMyMjQ2OTM1NCwKICAgICAgICAgICAgICAgICAgICAgICAiTG9uZ2l0dWRlRGVncmVlcyI6IDE1Ljk2M
DUzOTIyNTQ4ODkwMQogICAgICAgICAgICAgICAgICAgICAgICB9LAogICAgICAgICAgICAgICAgICAgICAgICAiQWx0aXR1ZGVNZXRlcnMiOiAxMjAsCiAgICAgICAgICAgICAgI
CAgICJEaXN0YW5jZU1ldGVycyI6IDcuMTE5OTk5ODg1NTU5MDgyLAogICAgICAgICAgICAgICAgICAgICAgICAiSGVhcnRSYXRlQnBtIjogewogICAgICAgICAgICAgI
CAgICAgICAgICJWYWx1ZSI6IDk0CiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgI
CAgICJUUFgiOiB7CiAgICAgICAgICAgICAgICAgICAgICAgICAiU3BlZWQiOiAxLjUwMTk5OTk3NDI1MDc5MzUsCiAgICAgICAgICAgICAgICAgICAgICAgICAiUnVuQ
2FkZW5jZSI6IDg3CiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIC
.2022. 21:02:45
```

Slika 27. Primjer ispisa kontrolnog dnevničkog zapisa stvarnovremenskog upravitelja sučelja

Na sličan način dohvaćaju se polustrukturirani i strukturirani podaci i pohranjuju u perzistentnu memoriju. Valja naglasiti da u slučaju strukturiranih podataka, stvarnovremenski upravitelj sučelja upućuje CQL upite prema sučelju strukturiranih podataka.

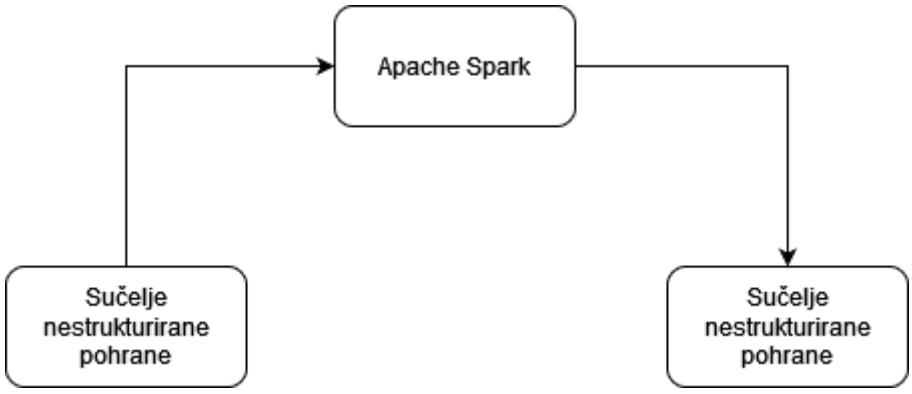
Unutar gotovo stvarnovremenskog podatkovnog cjevovoda verificira se algoritam 5, budući da je implementiran stvarnovremenski upravitelj sučelja čiji je navedeni algoritam esencijalni dio.

Apache Kafka implementira vlastiti privremeni spremnik podataka prikupljenih iz izvora podataka. Ujedno ima i ugrađen sustav za upravljanje zapisa u privremenom spremniku koji odgovara algoritmu 6.

### 5.3.3. Transformacijski podatkovni cjevovod

Transformacijski podatkovni cjevovod temeljen je na paralelnoj obradi podataka pohranjenih u zoni nepromijenjenih podataka korištenjem programskog alata Spark. Kao i stvarnovremenski podatkovni cjevovod, njegova konfiguracija počiva na metapodacima. Pritom su uz intraobjektne metapodatke korišteni i interobjektni metapodaci, iz razloga što se obavljaju transformacije podataka.

Izvor i odredište podataka transformacijskog podatkovnog cjevovoda je sloj pohrane podataka, kojem se pristupa kroz sučelja nestrukturirane pohrane, odnosno sučelje polustrukturirane pohrane. Arhitektura cjevovoda prikazana je na slici 28.




Slika 28. Arhitektura transformacijskog podatkovnog cjevovoda

U transformacijskom podatkovnom cjevovodu, ulogu upravitelja preslikavanja strukture podataka ima aplikacija Apache Spark. Apache Spark je u ovom slučaju odabran zbog mogućnosti jednostavnog izvršavanja zadataka u raspodijeljenoj okolini, što ima utjecaj na performanse zadataka preslikavanja.

U nastavku dan je prikaz metapodataka korištenih u podatkovnom cjevovodu u tablici 21. Pridruženi izvor podataka za entitete je jezero podataka, no kako se dohvaćaju iz različitih sučelja, označeni su s DL_N (nestrukturirani) i DL_P (polustrukturirani).

Između entiteta postoji transformacija, opisana metapodacima u tablici 22.

Tablica 21. Metapodaci entiteta u transformacijskom podatkovnom cjevovodu

| Kôd izvora podataka | Struktura | Kôd grupe izvora podataka |
|---|---|---|
| GPS_API_N | nestrukturirani | DL_N |
| MaximumHeartRateBpm | polustrukturirani | DL_P |
| AverageHeartRateBpm | polustrukturirani | DL_P |
| Track | polustrukturirani | DL_P |
| Trackpoint | polustrukturirani | DL_P |
| Lap | polustrukturirani | DL_P |
| Activities | polustrukturirani | DL_P |
| GPS_PowerBI_model | polustrukturirani | DL_P |

Metapodaci transformacije definiraju proceduru koju će Apache Spark pokrenuti u svrhu preslikavanja između nepromijenjenih podataka (u nestrukturiranom obliku) i podataka prilagođenih za vizualizacijski model u polustrukturiranom obliku. Navedeni metapodaci dani su kao primjer opisa transformacije u tablici 22.

Tablica 22. Metapodaci transformacije u transformacijskom podatkovnom cjevovodu

| Kôd izvornog entiteta | Kôd odredišnog entiteta | Kôd vrste odnosa entiteta | Procedura transformacije |
|---|---|---|---|
| GPS_API_N | GPS_PowerBI_model | IO | GPS_N_To_PowerBI.py |

Unutar procedure definirane u metapodacima transformacije eksplicitno dolazi do pretvaranja nestrukturiranog zapisa u niz atributa, koji se potom preslikavaju u polustrukturirane zapise na temelju pravila definiranih u proceduri. Metapodaci atributa u entitetima prikazani su u tablici 23.

Tablica 23. Metapodaci atributa u transformacijskom podatkovnom cjevovodu

| Kôd atributa | Naziv atributa | Kôd entiteta | Tip podataka |
|---|---|---|---|
| 15 | gps_data | GPS_API_N | blob |

| | | | |
|---|---|---|---|
| 16 | Track | Lap | object |
| 17 | Trackpoint | Track | object |
| 18 | Time | Trackpoint | timestamp |
| 19 | Position | Trackpoint | object |
| 20 | LatitudeDegrees | Position | float |
| 21 | LongitudeDegrees | Position | float |
| 22 | AltitudeMeters | Trackpoint | float |
| 23 | DistanceMeters | Trackpoint | float |
| 24 | HeartRateBpm | object | int |
| 25 | Value | HeartRateBpm | int |
| 26 | Extensions | Trackpoint | object |
| 27 | TPX | Extensions | object |
| 28 | Speed | TPX | float |
| 29 | RunCadence | TPX | int |

Unutar transformacijskog cjevovoda, u Python proceduri GPS_N_To_PowerBI.py implementiran je algoritam 4. Njegovom implementacijom vidljiva je potpora za prostorno-vremenske tipove podataka, iz razloga što je iz tipa podataka *blob* dobiven LPAV unutar kojeg postoje vremenska i prostorna domena. Dok je za vremensku domenu vidljivo kako se radi o atributu Time, prostorna domena sadržana je u atributima LatitudeDegrees i LongitudeDegrees. Iako se ne radi eksplicitno o prostornom tipu podataka, navedeni atributi dobiveni su unutar procedure iz prostornog tipa podataka *point*, nakon čega su prilagođeni za vizualizacijski model.

### 5.3.4. Vizualizacijski model

Vizualizacije prostorno-vremenskih tokova podataka izvedene su korištenjem alata PowerBI, opisanog ranije u poglavlju 4.5. Vizualizacijski model konzumira podatke iz zone aplikativnog pristupa, u kojoj su podaci oblikovani u modele pogodne za korištenje u pojedinim aplikacijama.

Korišteni podaci rezultat su stvarnovremenske obrade podataka u podatkovnom cjevovodu i povezanih transformacija koji su iz GPS izvora podataka i povezanih senzora stvorili polustrukturirani zapis.

Vizualizirani su podaci o tri različite rute dobivene iz tri različita izvora podataka (strukturiranom, polustrukturiranom i nestrukturiranom). Vizualizacijski model opisan je metapodacima u tablicama 24 (entiteti) i 25 (atributi).

Tablica 24. Metapodaci entiteta vizualizacijskog modela

| Kôd entiteta | Struktura |
|---|---|
| MaximumHeartRateBpm | polustrukturirani |
| AverageHeartRateBpm | polustrukturirani |
| Track | polustrukturirani |
| Trackpoint | polustrukturirani |
| Lap | polustrukturirani |
| Activities | polustrukturirani |
| GPS_PowerBI_model | polustrukturirani |

Tablica 25. Metapodaci atributa vizualizacijskog modela

| Kôd atributa | Naziv atributa | Kôd entiteta | Tip podataka |
|---|---|---|---|
| 30 | Activities | GPS_PowerBI_model | object |
| 31 | Author | GPS_PowerBI_model | object |
| 32 | Lap | Activities | object |
| 33 | @Sport | Lap | string |
| 34 | ID | Lap | timestamp |
| 35 | TotalTimeSeconds | Lap | int |
| 36 | DistanceMeters | Lap | int |
| 37 | MaximumSpeed | Lap | float |
| 38 | Calories | Lap | int |
| 39 | AverageHeartRateBpm | Lap | object |
| 40 | MaximumHeartRateBpm | Lap | object |
| 41 | Value | AverageHeartRateBpm | int |
| 42 | Value | MaximumHeartRateBpm | int |
| 43 | Intensity | Lap | string |
| 44 | TriggerMethod | Lap | string |
| 45 | Track | Lap | object |
| 46 | Trackpoint | Track | object |

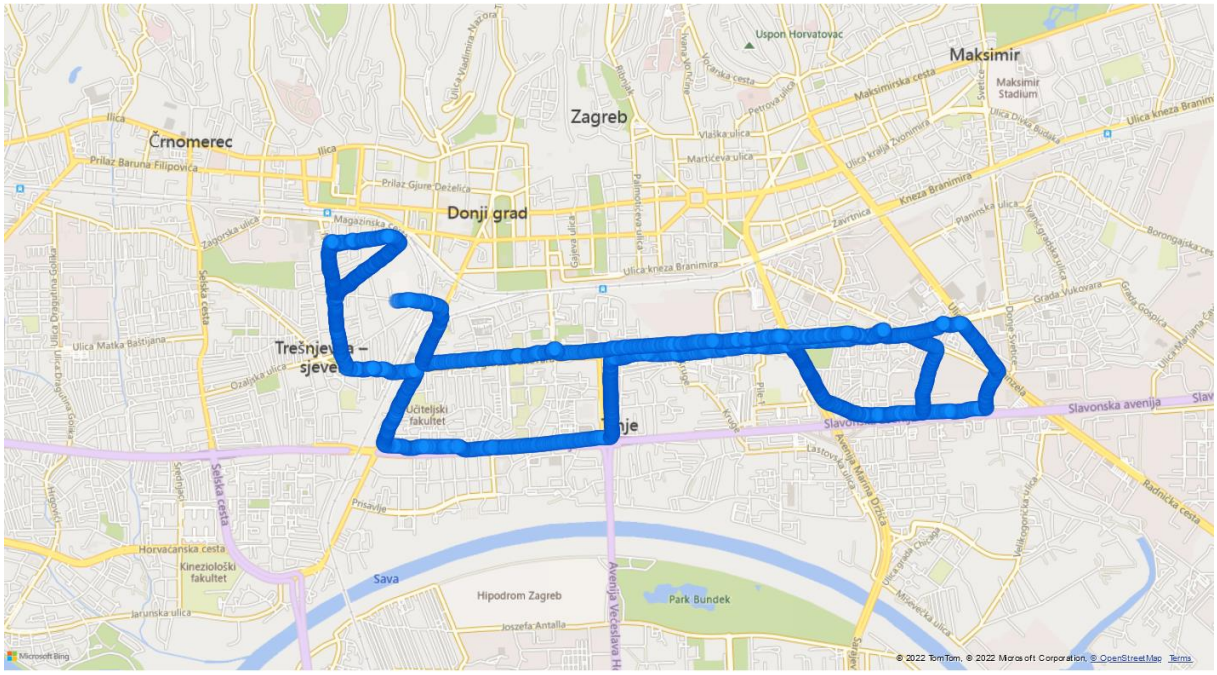
| | | | |
|---|---|---|---|
| 47 | Time | Trackpoint | timestamp |
| 48 | Position | Trackpoint | object |
| 49 | LatitudeDegrees | Position | float |
| 50 | LongitudeDegrees | Position | float |
| 51 | AltitudeMeters | Trackpoint | float |
| 52 | DistanceMeters | Trackpoint | float |
| 53 | HeartRateBpm | object | int |
| 54 | Value | HeartRateBpm | int |
| 55 | Extensions | Trackpoint | object |
| 56 | TPX | Extensions | object |
| 57 | Speed | TPX | float |
| 58 | RunCadence | TPX | int |


```
{
    "Activities": {
    "Activity": {
                "@Sport": "Running",
      "Id": "2022-04-27T16:05:06.000Z",
       "Lap": [
        {
          "TotalTimeSeconds": 60,
          "DistanceMeters": 142.97,
          "MaximumSpeed": 2.753000020980835,
          "Calories": 10,
          "AverageHeartRateBpm": {
            "Value": 131
          },
          "MaximumHeartRateBpm": {
            "Value": 151
          },
          "Intensity": "Active",
          "TriggerMethod": "Manual",
          "Track": {
            "Trackpoint": [
              {
                "Time": "2022-04-27T16:05:06.000Z",
                "Position": {
                  "LatitudeDegrees": 45.803568698465824,
                  "LongitudeDegrees": 15.960449958220124
                },
                "AltitudeMeters": 120.5999984741211,
                "DistanceMeters": 0.03999999910593033,
                "HeartRateBpm": {
                  "Value": 87
                },
                "Extensions": {
                  "TPX": {
                    "Speed": 0.0560000017285347,
                    "RunCadence": 0
                  }
                }
              },
```

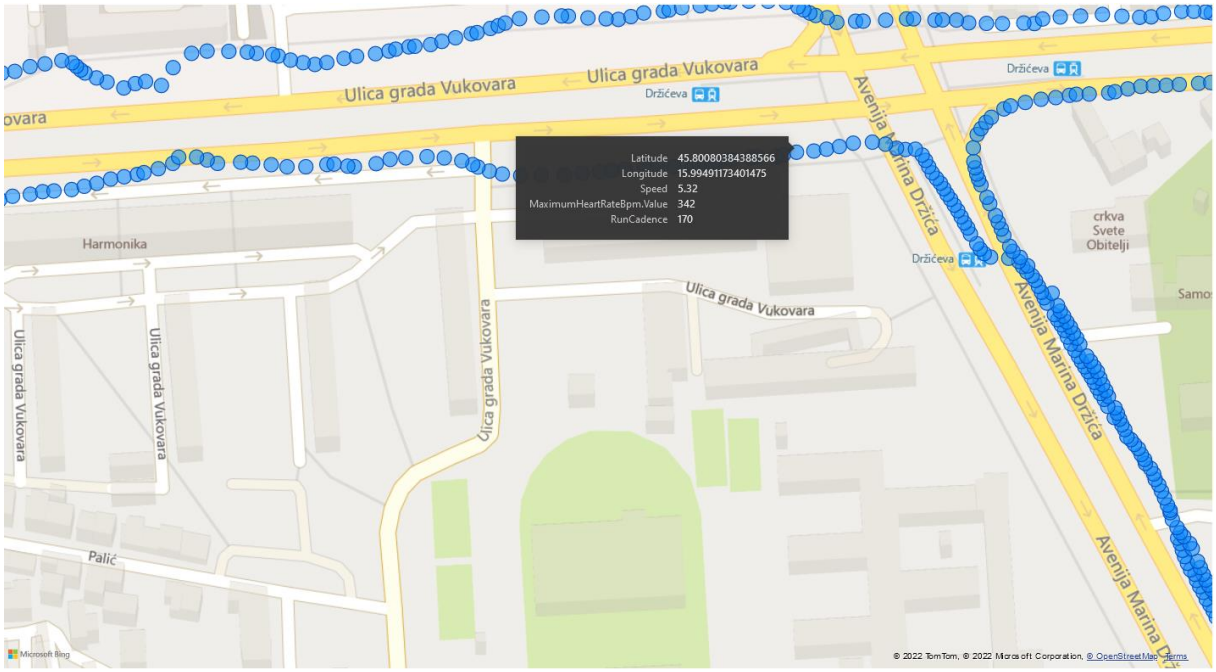

Slika 29. Primjer polustrukturiranog vizualizacijskog modela s GPS i senzorskim podacima

Slijedeći definiranu arhitekturu jezera podataka, svi progutani podaci su kroz podatkovni cjevovod preoblikovani u polustrukturirani vizualizacijski model. Jedinstveni vizualizacijski model odlika je zonske arhitekture jezera podataka i nalazi se u zoni aplikativnog pristupa. Primjer strukture zapisa vizualizacijskog modela prikazan je na slici 29.

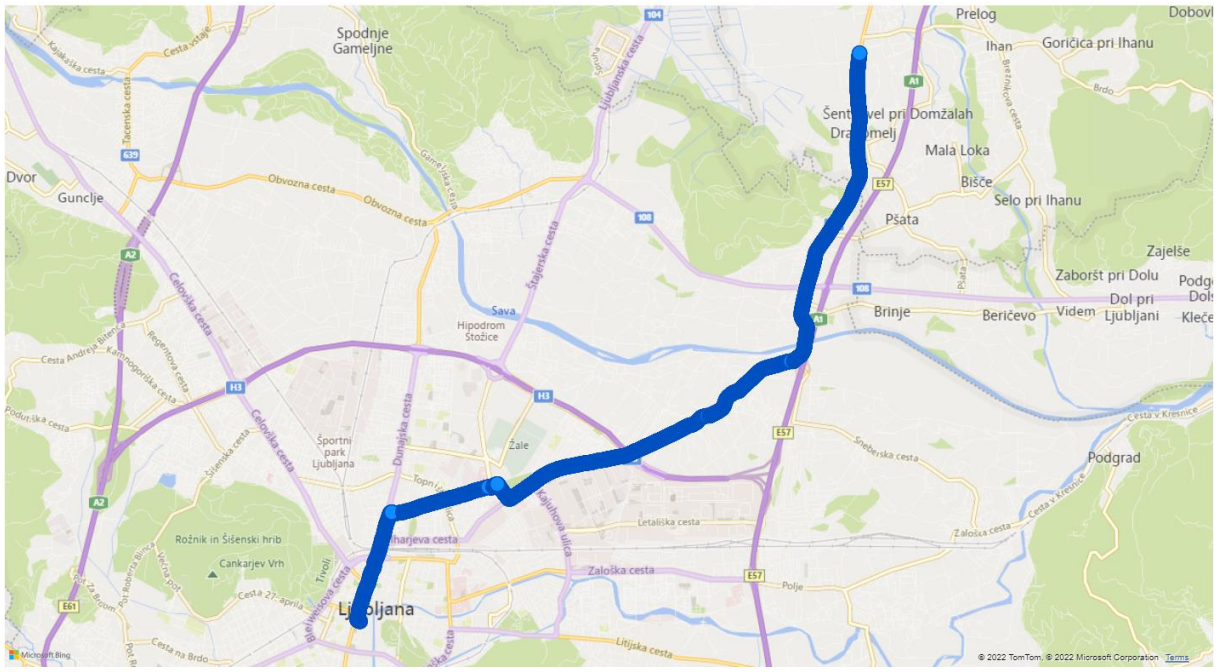
U nastavku su dani prikazi vizualizacija podataka dobivenih iz nestrukturiranih (slike 30 i 31), polustrukturiranih (slike 32 i 33) te strukturiranih (slike 34 i 35) izvora podataka, prilagođenih u vizualizacijski model.



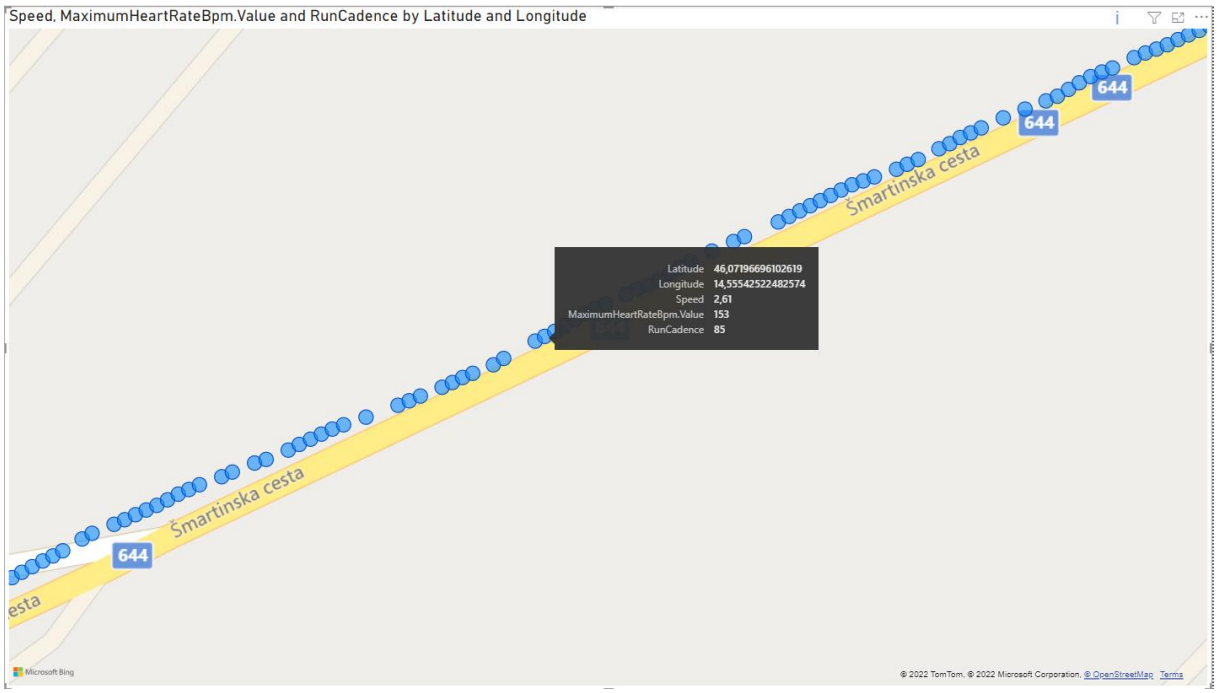

Slika 30. Vizualizacija nestrukturiranih prostorno-vremenskih podataka iz tokova podataka




Slika 31. Vizualizacija nestrukturiranih prostorno-vremenskih podataka iz tokova podataka

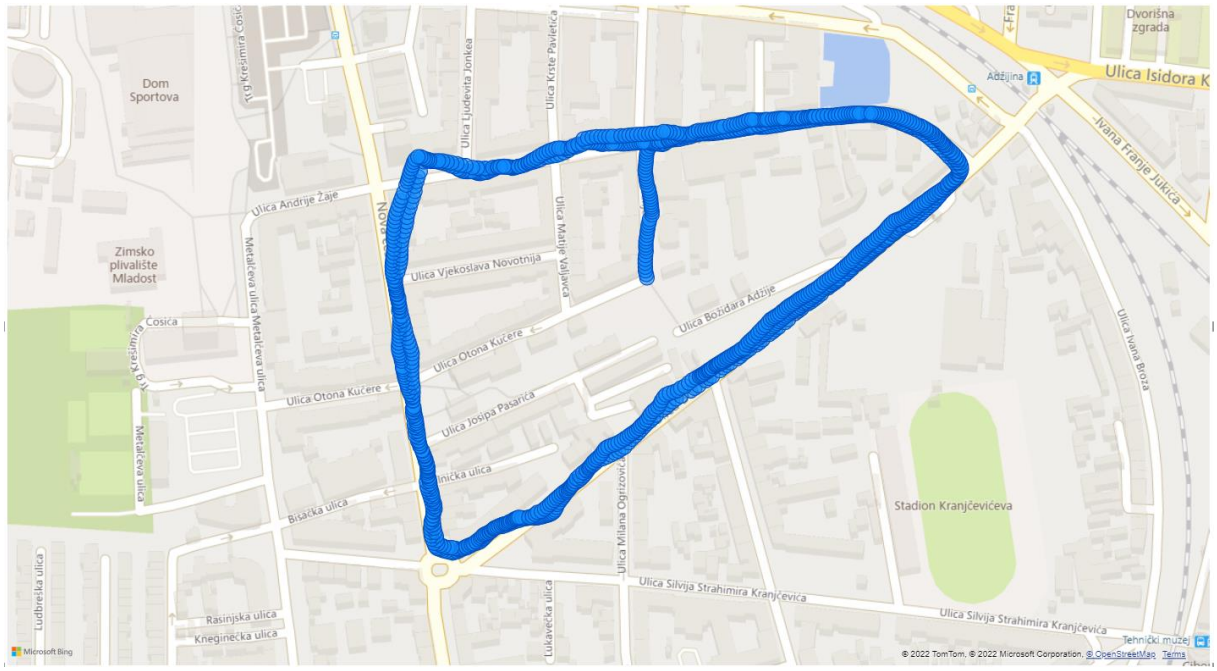



Slika 32. Vizualizacija polustrukturiranih prostorno-vremenskih podataka iz tokova podataka

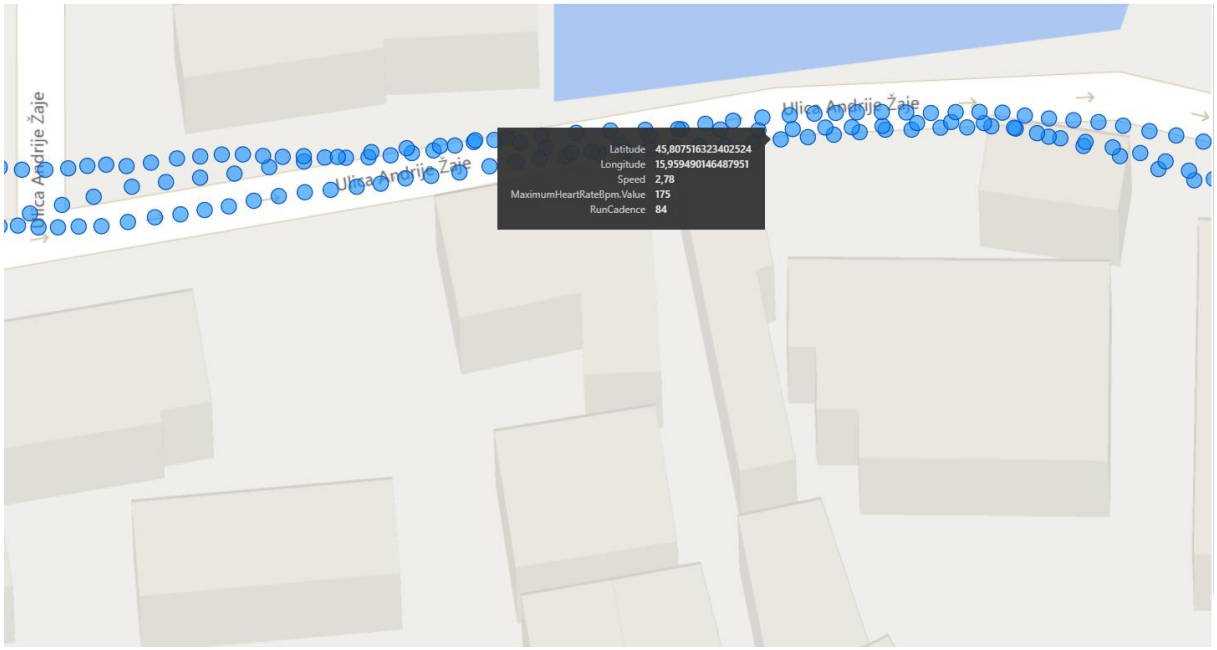



Slika 33. Vizualizacija polustrukturiranih prostorno-vremenskih podataka iz tokova podataka

Iako nije eksplicitno prikazano primjerima prilikom opisa transformacijskog cjevovoda, u njemu je u svrhu vizualizacije u zasebnoj proceduri GPS_S_To_PowerBI.py implementirano preslikavanje strukturiranih u polustrukturirane zapise. Kroz verifikaciju ispravnog prikaza preslikanih strukturnih podataka, ujedno je verificiran i algoritam 3 koji opisuje takvo preslikavanje.




Slika 34. Vizualizacija prilagođenih strukturiranih prostorno-vremenskih podataka iz tokova podataka




Slika 35. Vizualizacija prilagođenih strukturiranih podataka

### 5.3.5. Verifikacija modela - zaključak

Jezera podataka u početku su s implementacijskog gledišta bila smatrana isključivo sustavima temeljenima na platformi Apache Hadoop, što je djelomično proizašlo iz korištenja dvozonske arhitekture. Međutim, kroz razvoj arhitektura jezera podataka i preciznije definiranje njihove uloge u analitičkom sustavu, pokazala se potreba za implementacijom dodatnih sustava za pohranu podataka. Njihova je uloga, uz pohranu podataka u izvornom obliku, ostvarivanje optimalnog rada s odgovarajućim tipom podataka. Stoga se, uz nestrukturiranu pohranu podataka koju nudi Apache Hadoop, implementiraju i postrelacijske baze podataka te baze podatkaa prilagođene radu s polustrukturiranim podacima.

Uz različite sustave za pohranu podataka unutar istog jezera podataka, potrebno je osigurati sloj obrade podataka. U predloženom modelu jezera podataka, sloj obrade podataka implementiran je kao UDAL s pripadnim repozitorijem metapodataka. UDAL je skup više aplikacija koje imaju uloge prikupljanja podataka, njihove obrade i dinamičkog upravljanja izvođenjem aplikacija. Aplikacije pomoću kojih je izgrađen UDAL primarno se odlikuju podrškom za rad u raspodijeljenim sustavima, dok su preostale aplikacije prilagođene neovisnom izvođenju na različitim čvorovima unutar sustava.

Unutar UDAL ugrađena je potpora radu s prostorno-vremenskim tokovima podataka, primarno kroz ugradnju potpore za prostorno-vremenske tipove podataka. Generalno, potpora tokovima podataka ostvarena je putem sučelja i odgovarajućih upravitelja i aplikacija unutar podatkovnih cjevovoda. Implementirane aplikacije sadrže potporu za prostorno-vremenske tipove podataka i operacije nad njima, dok poneke dolaze s implementacijama algoritama.

U unificiranom sloju pristupa podacima implementirani su podatkovni cjevovodi s primarnom ulogom gutanja podataka, dok se u zonskoj arhitekturi ujedno koriste za integraciju podataka. Pritom se kao izvor podataka koristi perzistentna memorija, transformacije se obavljaju u podatkovnim cjevovodima, dok je odredište jedna od zona unutar jezera podataka sa specijaliziranom namjenom. Podatkovni cjevovodi u raspodijeljenim sustavima optimalno se stvaraju korištenjem aplikacija i sustava prilagođenih radu u raspodijeljenom okruženju, koji omogućuju jednostavnu skalabilnost, paralelne obrade zadataka i visoku dostupnost.

Pored korištenih aplikacija, za uspješnu implementaciju sloja obrade podataka, a time i uspješnu implementaciju jezera podataka, nužno je korištenje repozitorija metapodataka. U predloženom prototipu jezera podataka, repozitorij metapodataka osnova je izgradnje podatkovnih cjevovoda, koji su izgrađeni s pristupom parametrizacije pojedinih segmenata.

Pritom su parametri u cjevovodima sadržani u repozitoriju metapodataka u interobjektnim klasama, a UDAL i njegovi upravitelji zaduženi su za parametrizirano pozivanje aplikacija koje se nalaze unutar cjevovoda.

Algoritmi iz poglavlja 3.7 verificirani su kroz izgradnju podatkovnih cjevovoda i točne očekivane podatke pohranjene u perzistentnu memoriju. Dok je dio algoritama verificiran isključivo u samim podatkovnim cjevovodima, pojedini su algoritmi verificirani tek kroz vizualizacije podataka gdje su ispravno prikazani u krajnjoj vizualizacijskoj aplikaciji.

Odabir zonske arhitekture omogućio je međusobno neovisno korištenje različitih oblika istog zapisa u odvojene svrhe. Tako su nepromijenjeni zapisi dostupni kao izvor za prilagodbu strukture u svrhu naprednih analiza i vizualizacija, ali i u svrhu omogućavanja korisničkog pristupanja podacima u jasnoj strukturi. Primjer prilagodbe podataka vidljiv je u vizualizacijama, gdje su podaci različitih struktura pohranjeni u unificiranoj strukturi prilagođenoj izvještajnom modelu.

# 6. Zaključak

Jezera podataka nastala su kao odgovor na potrebe stvarnovremenske analize velike količine heterogenih podataka koji nastaju velikom brzinom, odnosno velikih podataka. Dotadašnji dominantni analitički sustavi, pretežno temeljeni na RSUBP, nisu bili prilagođeni takvim izazovima. Dok su takvi sustavi pronašli svoju ulogu u poslovnom okruženju koje zahtijeva konzistentnost podataka, kao što su potpora operativnom poslovanju i sustavi za potporu odlučivanju, veliki podaci i jezera podataka to proširuju kroz potporu za brzo, taktičko poslovno odlučivanje.

Veliki podaci, u poslovnom smislu, nastaju manjim dijelom unutar organizacije, a većim dijelom van nje u izvorima podataka poput društvenih mreža, mobilnih uređaja i različitih senzora. Upravo takvi podaci u sebi sadrže najviše informacija koje su bile teško dostupne ili čak potpuno nedostupne u poslovnom okruženju do pojave napredne analize i sustava za efikasni rad s velikim podacima.

Sve veći broj uređaja koji u sebi sadrže senzore, što uključuje i pokretne senzore, stvara tokove podataka koji uz vremenske sadrže i prostorne podatke. Iako su prethodno prostorni podaci dobro definirani, u njima je izostala potpora za nestrukturirane tipove podataka. Iz tog razloga, postojeća istraživanja i formalne okvire nužno je proširiti. U ovom radu, kao jedan od doprinosa, adresirana je navedena problematika kroz dodavanje potpore nestrukturiranim tipovima podataka u postojeći skup podržanih tipova podataka. Uz to, postojeći operatori nad tipovima podataka prošireni su podrškom za rad s nestrukturiranim tipovima podataka.

Jezera podataka, uzevši u obzir njihovu sve veću rasprostranjenost, prestala su biti samo repozitorij nepromijenjenih podataka te su preuzela i ulogu izvora za analitičke upite korisnika, alate za naprednu analizu, ali i za vizualizacijske alate. Pritom je razvoj alata, koji su sve više počeli uvoditi potporu za polustrukturirane i nestrukturirane podatke, odigrao važnu ulogu jer je struktura podataka prestala biti prepreka za njihovo korištenje nakon pohrane. Istovremeno, arhitekture jezera podataka počele su se razvijati kako bi se ona oblikovala prema korporativnim standardima i podigla svoju razinu zrelosti.

S vremenom se pojavilo nekoliko različitih pristupa stvaranju modela arhitekture jezera podataka, od kojih su se u analizi najzrelijima i najkompletnijima pokazali pristupi izgradnje zonskog logičkog modela i slojevitog modela. Budući da zonska arhitektura ne definira načine obrade podataka, već samo logički strukturni model nad pohranjenim podacima, u svrhu sveobuhvatne definicije arhitekture jezera podataka potrebno je definirati i slojeve jezera

podataka u kojima se podaci obrađuju i pohranjuju. Spajanjem definicija zonske i slojevite arhitekture, dobivena je predložena arhitektura jezera podataka koja sadrži šest logičkih zona podataka, sloj pohrane podataka i sloj obrade podataka. Dok je po pitanju zona predložena arhitektura bliska postojećim zonskim arhitekturama, slojevita arhitektura značajno je dorađena. Pretežno se dorada odnosi na preklapanje uloga pojedinih slojeva i zona, koje su potom definirane kao dio zone. Ujedno, definiranjem sloja obrade podataka u jezeru podataka stvoreni su preduvjeti za izradu modela unificiranog sloja pristupa podacima – UDAL.

U predloženom modelu jezera podataka, zone prilagođenih podataka i prilagođenog pristupa podatke pohranjuju u polustrukturiranom zapisu. Polustrukturirani zapis odabran je iz nekoliko razloga. S tehničke strane, dovoljno je raširena potpora za rad s polustrukturiranim podacima u aplikacijama za naprednu analizu i vizualizacijskim alatima, a čak su i pojedini RSUBP ugradili potporu za njih. Zatim, preslikavanje podataka iz strukturiranog oblika u polustrukturirani značajno je jednostavnije od preslikavanja iz polustrukturiranog u strukturirani, pogotovo u slučaju ugniježđenih hijerarhija. Jednako tako, preslikavanje nestrukturiranih zapisa u strukturirani nije nužno podržano u svim RSUBP, dok polustrukturirani zapisi nemaju takvo ograničenje.

Doprinos ovog rada je definicija unificiranog sloja pristupa podacima unutar jezera podataka s ugrađenom potporom za rad s prostorno-vremenskim tokovima podataka. Model UDAL definiran je kao međusloj unutar jezera podataka s višestrukim zadaćama. Najčešće su među njima zadaće gutanja i integracije podataka kroz podatkovne cjevovode izgrađene dinamički na temelju metapodataka. Zatim, UDAL ima zadaću dohvata i prikaza podataka iz perzistentne pohrane u jezeru podataka u svrhu potpore analitičkim upitima i aplikacijama. Konačno, UDAL pruža mogućnost pristupa dohvaćenim podacima koji još nisu pohranjeni u perzistentnu memoriju u svrhu stvarnovremenske analize sa što manjom latencijom.

UDAL se sastoji od skupa ulaznih sučelja, skupa izlaznih sučelja i skupa upravitelja koji zajedno tvore podatkovne cjevovode. Ulazna sučelja UDAL imaju namjenu prihvata podataka, dok izlazna sučelja imaju zadaću komunikacije sa slojem pohrane podataka, odnosno s vanjskim aplikacijama koje konzumiraju podatke. U podatkovnim se cjevovodima sučelja mogu ulančavati, tako da izlazno sučelje može biti ujedno izvor podataka za ulazno sučelje pri dohvatu podataka iz sloja pohrane podataka. Upravitelji u UDAL imaju raznolike uloge, od upravljanja različitim zadaćama do integriranja podataka i promjene strukture. Unutar jednog podatkovnog cjevovoda u UDAL nalaze se minimalno jedno ulazno sučelje, minimalno jedno

izlazno sučelje te niz upravitelja koji upravljaju podacima u samom podatkovnom cjevovodu. Potpora tokovima podataka očituje se u samim sučeljima, gdje su pojedina sučelja prilagođena radu s tokovima podataka kroz definirane periode osvježavanja, načine prihvata podataka u vidu potpore metodama guranja i povlačenja podataka iz vanjskih izvora.

Repozitorij metapodataka neodvojiv je dio sloja obrade podataka, budući da se cjelokupni UDAL naslanja na njega u različitim ulogama. Metapodaci su jedan od temelja ugradnje procesa upravljanja podacima, a u UDAL model su ujedno i osnova izgradnje cjevovoda. Metapodaci u repozitoriju dijele se na klase intraobjektnih metapodataka (koje opisuju strukturu podataka do razine atributa), klase interobjektnih metapodataka (koji opisuju odnose među izvorima podataka, entitetima i atributa), klase obrada i klase evolucija metapodataka. Održavani repozitorij metapodataka, osim što omogućava implementaciju UDAL, ujedno i daje mogućnosti praćenja porijekla podataka, održavanje sigurnosnih metapodataka za potporu zoni prilagođenog pristupa te pružanje uvida u promjene struktura nad podacima kroz pojedine zone.

U repozitoriju metapodataka, potpora prostorno-vremenskim tokovima podataka dana je kroz ugradnju specifičnih atributa u interobjektne klase i enumeracije izvora podataka, gdje su tokovi podataka posebno označeni.

Unutar UDAL ugrađeni su algoritmi potpore operacijama nad prostorno-vremenskim tipovima podataka. Navedeni algoritmi definirani su na apstraktnoj razini i podijeljeni u dvije grupe: skup algoritama za spajanje tokova podataka te skup algoritama za potporu upraviteljima UDAL, koji uključuju operacije preslikavanja struktura podataka. Ugradnjom algoritma dobiven je jednoznačan pristup obavljaju pojedinih operacija na razini cijelog UDAL.

Razvijeni model jezera podataka, uz definirani UDAL, daje cjelokupni pogled na potrebne komponente jezera podataka kako bi se ono promatralo kao jedinstveni analitički sustav. U prethodnim definicijama fokus je dio na domeni pohrane ili domeni obrade podataka, no nijedno istraživanje nije uzimalo oba segmenta rada s podacima u obzir. Definiranje UDAL kao jedinstvenog međusloja za obradu i dohvat podataka temeljenog na repozitoriju metapodataka podloga je za implementaciju sloja obrade podataka koji se ne temelji isključivo na jednoj tehnologiji ili programskom okviru.

Prijedlog proširenog skupa tipova podataka i operatora, zajedno s prijedlogom korištenja polustrukturirane strukture zapisa na razini cijelog jezera podataka u svrhu ujednačavanja strukture, značajan je zbog smjera odmaka od strukturiranih zapisa korištenih u svrhu potpore za analitičke aplikacije. Ujedno je fokus na prostorno-vremenske tokove podataka kao trenutno

iznimno značajne izvore podataka postojeća istraživanja i formalne okvire primaknuo trendovima u analitičkom svijetu.

### Pretpostavke i ograničenja

Prva pretpostavka jest da svi odabrani sustavi za pohranu podataka imaju potporu za rad s prostorno-vremenskim podacima ili je mogu ostvariti putem odgovarajućih proširenja. Predložena arhitektura jezera podataka temelji se na potpori za tri vrste struktura podataka te potpora ni za koju ne smije izostati.

Zatim, svi sustavi za pohranu podataka moraju posjedovati neki oblik sučelja pomoću kojeg je moguće ostvariti komunikaciju s njima. Sučelje pohrane podataka ne mora biti nužno aplikacijsko programsko sučelje, može biti i sučelje naredbene linije (eng. *command line interface*).

Prethodno je istaknuto ograničenje u formalnoj definiciji skupa nosioca za tip podataka *blob* na tekstualne zapise. Implementacija tipa podataka *blob*, iako označava nestrukturirani podatak, varira od sustava do sustava. U slučajevima da je kao *blob* sadržan, primjerice, slikovni ili video zapis, očekuje se da će biti reprezentiran znakovima prikazanima u formalnoj definiciji danom u ovom radu.

### Buduća istraživanja

Jedan od fokusa budućih istraživanja svakako će biti u domeni automatske ekstrakcije metapodataka iz izvornih zapisa i ugradnji takvog sustava u UDAL. Izvori podataka nemaju uvijek unaprijed poznatu strukturu koja bi se unijela u repozitorij metapodataka, a pritom bi automatsko rješenje smanjilo potrebu za ručnom intervencijom i omogućilo efikasniji rad s podacima. Postojeća rješenja potrebno je analizirati i odlučiti koja su pogodna za ugradnju u UDAL i podatkovne cjevovode.

Također, pažnja će se u budućim istraživanjima pridodati ugradnji sustava za spajanje entiteta (eng. e*ntity matching*). Procesi spajanja entiteta iznimno su važni u velikim podacima, budući da se pomoću njih mogu jednostavno povezati različiti tokovi podataka. Do izražaja takvi procesi dolaze prilikom rada s nestrukturiranim podacima, pri čemu klasifikacija zapisa može doprinijeti većoj razini kvalitete pohranjenih podataka u jezerima podataka.

Drugi smjer istraživanja očekuje se u razvoju novih modela jezera podataka. Pritom će se velika pažnja posvetiti arhitekturi jezerskog skladišta podataka, ali i odabiru prikladnih arhitektura ovisno o njegovom smještaju (eng. *deployment*). U posljednje vrijeme veliki fokus

je na tehnologijama u oblaku, ali i hibridnom pristupu smještaju kao poluge između postojećih lokalnih smještaja i smještaja u oblaku.

# Reference


[1] Miloslavskaya, N., & Tolstoy, A. (2016). Big Data, Fast Data and Data Lake Concepts. *Procedia Computer Science*, *88*, 300–305.

[2] Golov, N., & Rönnbäck, L. (2015). Big Data normalization for massively parallel processing databases. Computer Standards and Interfaces, 54, 86–93.

[3] Raman, K., Swaminathan, A., Gehrke, J., & Joachims, T. (2013). Beyond myopic inference in big data pipelines. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Part F128815, 86–94.

[4] Sawadogo, P., & Darmont, J. (2021). On data lake architectures and metadata management. Journal of Intelligent Information Systems, 56(1), 97–120.

[5] Dixon, J. (2010). Pentaho, Hadoop, and Data Lakes. https://jamesdixon.wordpress.com/2010/10/14/pentaho-hadoop-and-data-lakes/. Datum pristupa: 28. 10. 2021

[6] Fang, H. (2015). Managing data lakes in big data era: What's a data lake and why has it became popular in data management ecosystem. *2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*, 820–824.

[7] Madera, C., & Laurent, A. (2016). The next information architecture evolution: The data lake wave. *8th International Conference on Management of Digital EcoSystems, MEDES 2016*, 174–180.

[8] Sawadogo, P. N., Scholly, É., Favre, C., Ferey, É., Loudcher, S., & Darmont, J. (2019). Metadata Systems for Data Lakes: Models and Features. *Communications in Computer and Information Science*, *1064*, 440–451.

[9] Yebenes, J., & Zorrilla, M. (2019). Towards a data governance framework for third generation platforms. *Procedia Computer Science*, *151*, 614–621.

[10] Weber, K., Otto, B., & Österle, H. (2009). One Size Does Not Fit All---A Contingency Approach to Data Governance. *Journal of Data and Information Quality*, *1*(1), 1–27.

[11] Data Administration Management Association, „DAMA-DMBOOK: Data Management Body of Knowledge“, Technics Publications, 2009

[12] Data Administration Management Association, „DAMA-DMBOOK2: Data Management Body of Knowledge“, Technics Publications, 2015

[13] Majid Al-Ruithe, Development and Evaluation of a Holistic Framework and Maturity Assessment Tools for Data Governance in Cloud Computing Environments, (Staffordshire University, 2018).

[14] Suriarachchi, I., & Plale, B. (2017). Crossing analytics systems: A case for integrated provenance in data lakes. Proceedings of the 2016 IEEE 12th International Conference on E-Science, e-Science 2016, 349–354.

[15] Brackenbury, W., Liu, R., Mondal, M., Elmore, A. J., Ur, B., Chard, K., & Franklin, M. J. (2018). Draining the data swamp: A similarity-based approach. Proceedings of the Workshop on Human-In-the-Loop Data Analytics, HILDA 2018.

[16] Lampson, B. W. (1983). Hints for computer system design. Proceedings of the 9th ACM Symposium on Operating Systems Principles, SOSP 1983, 33–48.

[17] Lin, J. (2017). The Lambda and the Kappa. IEEE Internet Computing, 21(5), 60–66.

[18] Feick, M., Kleer, N., & Kohn, M. (2018). Fundamentals of Real-Time Data Processing Architectures Lambda and Kappa. SKILL 2018-Studierendenkonferenz Informatik, 1.

[19] Hasani, Z., Kon-Popovska, M., & Velinov, G. (2014). Lambda Architecture for Real Time Big Data Analytic. ICT Innovation, 133–143.

[20] Kroß, J., Brunnert, A., Prehofer, C., Runkler, T. A., & Krcmar, H. (2015). Stream processing on demand for lambda architectures. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 9272, 243–257.

[21] Kiran, M., Murphy, P., Monga, I., Dugan, J., & Baveja, S. S. (2015). Lambda architecture for cost-effective batch and speed big data processing. Proceedings - 2015 IEEE International Conference on Big Data, IEEE Big Data 2015, 2785–2792 .

[22] Munshi, A. A., & Mohamed, Y. A. R. I. (2018). Data Lake Lambda Architecture for Smart Grids Big Data Analytics. IEEE Access, 6(c), 40463–40471.

[23] Solodovnikova, D., Niedrite, L., & Niedritis, A. (2019). *On Metadata Support for Integrating Evolving Heterogeneous Data Sources*. New Trends in Databases and Information Systems. ADBIS 2019. Communications in Computer and Information Science, vol 1064. Springer, Cham, 378–390.

[24] Bilalli, B., Abelló, A., Aluja-Banet, T., & Wrembel, R. (2016). Towards intelligent data analysis: The metadata challenge. *IoTBD 2016 - Proceedings of the International Conference on Internet of Things and Big Data*, *IoTBD*, 331–338.

[25] Diamantini, C., Giudice, P. Lo, Musarella, L., Potena, D., Storti, E., & Ursino, D. (2018). A new metadata model to uniformly handle heterogeneous data lake sources. In *Communications in Computer and Information Science* (Vol. 909). Springer International Publishing.

[26] Quix, Christoph; Hai, Rihan; Vatov, I. (n.d.). GEMMS: A Generic and Extensible Metadata Management System for Data Lakes.

[27] Kimball, R., & Ross, M. (2002). The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling (2nd ed.). John Wiley & Sons, Inc.

[28] Ravat, F., & Zhao, Y. (2019). *Metadata Management for Data Lakes*. New Trends in Databases and Information Systems. ADBIS 2019. Communications in Computer and Information Science, vol 1064. Springer, Cham, 37–44.

[29] Theodorou, V., & Diamantopoulos, N. (2019). GLT: Edge gateway ELT for data-driven intelligence placement. *Proceedings - 2019 IEEE/ACM Joint 4th International Workshop on Rapid Continuous Software Engineering and 1st International Workshop on Data-Driven Decisions, Experimentation and Evolution, RCoSE/DDrEE 2019*, 24–27.

[30] Maccioni, A., & Torlone, R. (2017). Crossing the finish line faster when paddling the Data Lake with kayak. Proceedings of the VLDB Endowment, 10(12), 1853–1856.

[31] Güting, R. H. (1993). Second-Order Signature: A Tool for Specifying Data Models, Query Processing, and Optimization. ACM SIGMOD Record, 22(2), 277–286.

[32] Križanović, K., „Prostorno-vremensko proširenje sustava za upravljanje tokovima podataka“, doktorska disertacija, Fakultet elektrotehnike i računarstva, Zagreb, Sveučilište u Zagrebu, 2013.

[33] Križanović, K., Galić, Z., & Baranović, M. (2011). Data types and operations for spatio-temporal data streams. Proceedings - IEEE International Conference on Mobile Data Management, 2, 11–14

[34] Galić, Z., Baranović, M., Križanović, K., & Mešković, E. (2014). Geospatial data streams: Formal framework and implementation. Data and Knowledge Engineering, 91, 1–16.

[35] Mešković, E., Osmanović, D., Galić, Z., & Baranović, M. (2014). Generating spatio-temporal streaming trajectories. 2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2014 - Proceedings, May, 1130–1135.

[36] Galić, Z., Mešković, E., & Osmanović, D. (2017). Distributed processing of big mobility data as spatio-temporal data streams. GeoInformatica, 21(2), 263–291.

[37] Sveen, A. F. (2019). Efficient storage of heterogeneous geospatial data in spatial databases. Journal of Big Data, 6(1).

[38] Santos, P. O., Moro, M. M., & Davis, C. A. (2015). Comparative performance evaluation of relational and NOSQL databases for spatial and mobile applications. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 9261, 186–200.

[39] Eisenberg, A., & Melton, J. (1999). SQL:1999, formerly known as SQL3. SIGMOD Record (ACM Special Interest Group on Management of Data), 28(1), 131–138.

[40] Inmon, W. H. (2016). Data Lake Architecture: Designing the Data Lake and Avoiding the Garbage Dump. Technics Publications

[41] Quix, C., & Hai, R. (2019). Data Lake. Encyclopedia of Big Data Technologies, 552–559

[42] Sharma, B. (2018). Architecting Data Lakes, 2nd Edition. O'Reilley Media.

[43] Giebler, C., Gröger, C., Hoos, E., Eichler, R., Schwarz, H., & Mitschang, B. (2021). The Data Lake Architecture Framework: A Foundation for Building a Comprehensive Data Lake Architecture. Proceedings Der 19. Fachtagung Für Datenbanksysteme Für Business, Technologie Und Web (BTW 2021), 351–370.

[44] Munshi, A. A., & Mohamed, Y. A. R. I. (2018). Data Lake Lambda Architecture for Smart Grids Big Data Analytics. IEEE Access, 6, 40463–40471

[45] Seo, D. Y., Lee, D. H., Moon, K. S., Chang, J., Lee, J. Y., & Han, C. Y. (1997). Schemaless representation of semistructured data and schema construction. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 1308, 387–396.

[46] Rafique, A., Van Landuyt, D., Lagaisse, B., & Joosen, W. (2018). On the Performance Impact of Data Access Middleware for NoSQL Data Stores A Study of the Trade-Off between Performance and Migration Cost. IEEE Transactions on Cloud Computing, 6(3), 843–856.

[47] Inmon, W. H. (2002). Building the Data Warehouse,3rd Edition (3rd ed.). John Wiley & Sons, Inc.

[48] Eldawy, A., & Mokbel, M. F. (2015). SpatialHadoop: A MapReduce framework for spatial data. Proceedings - International Conference on Data Engineering, 2015-May, 1352–1363.

[49] Alarabi, L., & Mokbel, M. F. (2017). A demonstration of ST-Hadoop: A MapReduce framework for big spatio-temporal data. Proceedings of the VLDB Endowment, 10(12), 1961–1964.

[50] Butler, et. al., The GeoJSON format., https://datatracker.ietf.org/doc/html/rfc7946, Datum pristupa: 2. 5. 2021,

[51] da Costa Rainho, F., & Bernardino, J. (2018). Web GIS: A new system to store spatial data using GeoJSON in MongoDB. 2018 13th Iberian Conference on Information Systems and Technologies (CISTI), 1–6.

[52] Magic Quadrant for Data Science and Machine Learning Platforms, https://www.gartner.com/doc/reprints?id=1-29G5YKBI&ct=220321&st=sb, datum pristupa 07. 05. 2022

[53] Alteryx Spatial, https://help.alteryx.com/20214/designer/spatial, datum pristupa 07. 05. 2022

[54] Geospatial Analytics, https://knowledge.dataiku.com/latest/kb/analytics-ml/geospatial/index.html, datum pristupa 07. 05. 2022

[55] SAS/STAT Spatial Analysis Procedures: https://support.sas.com/rnd/app/stat/procedures/SpatialAnalysis.html, datum pristupa 07. 05. 2022

[56] Spatial Functions: https://docs.tibco.com/pub/spotfire/6.5.2/doc/html/ncfe/ncfe_spatial_functions.htm, datum pristupa 07. 05. 2022

[57] Hagedorn, S., Götze, P., & Sattler, K. U. (2017). The STARK framework for spatio-temporal data analytics on spark. Lecture Notes in Informatics (LNI), Proceedings - Series of the Gesellschaft Fur Informatik (GI), 265, 123–142.

[58] You, S., Zhang, J., & Gruenwald, L. (2015). Large-scale spatial join query processing in Cloud. Proceedings - International Conference on Data Engineering, 2015-June, 34–41.

[59] Yu, J., Wu, J., & Sarwat, M. (2015). GeoSpark: A cluster computing framework for processing large-scale spatial data. GIS: Proceedings of the ACM International Symposium on Advances in Geographic Information Systems, 03-06-Nove(3).

[60] Apache Sedona, https://sedona.apache.org/, datum pristupa 08. 05. 2022

[61] Orescanin, D., & Hlupic, T. (2021). Data Lakehouse - a Novel Step in Analytics Architecture. *2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO)*, 1242–1246.

[62] GeoPandas, https://geopandas.org/en/stable/index.html, datum pristupa 08. 05. 2022

[63] SPTEMP, https://pypi.org/project/sptemp/, datum pristupa 08. 05. 2022

[64] Graser, A. (2019). MovingPandas: Efficient structures for movement data in Python. GI_Forum, 7(1), 54–68.

[65] Rey, S. J. & Anselin, L. PySAL : A Python Library of Spatial Analytical Methods. Handbook of Applied Spatial Analysis, 37(2007), 175–193.

[66] PyMove, https://pymove.readthedocs.io/en/latest/, datum pristupa 09. 05. 2022

[67] SpatioTemporal: Spatio-Temporal Model Estimation, https://rdrr.io/cran/SpatioTemporal, datum pristupa 09. 05. 2022

[68] STMedianPolish, https://github.com/wamartinez/STMedianPolish, datum pristupa 09. 05. 2022

[69] Tesileanu, R. (2017). Geospatial analysis in Scala Geospatial analysis in Scala.

[70] Hughes, J. N., Annex, A., Eichelberger, C. N., Fox, A., Hulbert, A., & Ronquest, M. (2015). GeoMesa: a distributed architecture for spatio-temporal fusion. Geospatial Informatics, Fusion, and Motion Video Analytics V, 9473, 94730F.

[71] Eldawy, A., Sabek, I., Elganainy, M., Bakeer, A., Abdelmotaleb, A., & Mokbel, M. F. (2017). Sphinx: Empowering impala for efficient execution of SQL queries on big spatial data. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 10411 LNCS(1), 65–83.

[72] Geo-spatial Queries with Hive using ESRI Geometry and Spatial Framework for Hadoop, https://community.cloudera.com/t5/Community-Articles/Geo-spatial-Queries-with-Hive-using-ESRI-Geometry-and/ta-p/246737, datum pristupa 09.05.2022

[73] Map time-aware data, https://doc.arcgis.com/en/power-bi/design/map-time-aware-data.htm, datum pristupa 10.05.2022

[74] GeoAnalytics example: Drill-down map, https://help.qlik.com/en-US/geoanalytics/Content/Qlik_GeoAnalytics/GeoAnalytics-Sense/GeoAnalytics-ex-drilldown-map.htm, datum pristupa 10.05.2022

[75] Spatial Functions, https://help.tableau.com/current/pro/desktop/en-us/functions_functions_spatial.htm, datum pristupa 10.05.2022

[76] GeoLoop, https://grafana.com/grafana/plugins/citilogics-geoloop-panel/, datum pristupa 10.05.2022

# Životopis

Tomislav Hlupić završio je preddiplomski studij računarstva na Fakultetu elektrotehnike i računarstva Sveučilišta u Zagrebu i stekao prvostupničku diplomu 2012. godine, nakon čega nastavlja diplomski studij na FER-u te 2015. stječe titulu magistra inženjera informacijske i komunikacijske tehnologije. U veljači iduće godine upisuje doktorski studij računarstva na FER-u. Poslovno iskustvo u programiranju stjecao je već kao student radom na izradi aplikacija na nekolicini projekata. Po završetku diplomskog studija zapošljava se kao razvojni programer te radi na razvoju i održavanju sustava za korporativnu sigurnost. Od 2016. zaposlen je u tvrtki Poslovna inteligencija, gdje je trenutno na poziciji starijeg konzultanta. U dosadašnjoj konzultantskoj karijeri radio je na domaćim i inozemnim projektima pretežno u bankarskom i retail sektoru s fokusom na poslovnu inteligenciju i podatkovnu znanost. Od 2018. godine asistent je na Visokom učilištu Algebra gdje kasnije postaje predavač i nositelj nekoliko kolegija, a od 2020. radi i kao vanjski suradnik na Zavodu za primjenjeno računarstvo na FER-u. Paralelno uz poslovno iskustvo gradi znanstvenu karijeru, a područja istraživanja su mu skladišta podataka i poslovna inteligencija, jezera podataka, podaci pokretnih senzora i IPTV. Autor je osam članaka u časopisima i zbornicima konferencija.

# Popis radova

## Članci u časopisima

T. Hlupić, D. Oreščanin and M. Baranović, "**A Novel Method for IPTV Customer Behavior Analysis Using Time Series**," in *IEEE Access*, vol. 10, pp. 37003-37015, 2022, doi: 10.1109/ACCESS.2022.3164409.

## Članci u zbornicima konferencija

T. Hlupić, D. Oreščanin, D. Ružak and M. Baranović, "**An Overview of Current Data Lake Architecture Models**," 2022 45th International Convention on Information, Communication and Electronic Technology (MIPRO), 2022, pp. 1235-1240 (u procesu izdavanja)

D. Oreščanin and T. Hlupić, "**Data Lakehouse - a Novel Step in Analytics Architecture**," 2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO), 2021, pp. 1242-1246, doi: 10.23919/MIPRO52101.2021.9597091.

T. Hlupić and J. Puniš, "**An Overview of Current Trends in Data Ingestion and Integration**," 2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO), 2021, pp. 1265-1270, doi: 10.23919/MIPRO52101.2021.9597149.

Šušnjara, I.K., Hlupić, T. (2021). Using Hadoop Ecosystem and Python to Explore Climate Change. In: Bhattacharyya, S., Mršić, L., Brkljačić, M., Kureethara, J.V., Koeppen, M. (eds) Recent Trends in Signal and Image Processing. ISSIP 2020. Advances in Intelligent Systems and Computing, vol 1333. Springer, Singapore. https://doi.org/10.1007/978-981-33-6966-5_11

T. Hlupić, D. Oreščanin and A. -M. Petric, "Time series model for sales predictions in the wholesale industry," 2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO), 2020, pp. 1263-1267, doi: 10.23919/MIPRO48935.2020.9245255.

D. Oreščanin, T. Hlupić and I. Sorić, "Predictive models for digital broadcasting recommendation engine," 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2018, pp. 1243-1248, doi: 10.23919/MIPRO.2018.8400225.

T. Hlupić, F. Jandrijević, J. Kovačev, L. Petricioli, T. Gracin and M. Baranović, "System for monitoring and advanced analysis of handball matches," *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2015, pp. 1428-1433, doi: 10.1109/MIPRO.2015.7160498.

# Biography

Tomislav Hlupić finished undergraduate programme at the Faculty of Electronics and Computing at the University of Zagreb and in 2012 he earned his bachelor's degree in computing. He started master programme in the following year and in 2015 he was awarded Master of Science degree in Information and Communication Technology. Next academic year he enrolled in doctoral study programme of Electrical Engineering and Computing. He started gaining professional experience during his university days by working on several projects as a developer. After graduation he continued his carrer by working on development and maintance of corporate security systems. Since 2016 he has been working in Poslovna inteligencija, where he is currently holding a role of a senior consultant. Throughout his career he worked on domestic and international projects mainly in banking and retail industries with the focus on BI and data science. In 2018 he became a teaching assistant at Algebra University College and later he became a professor and course coordinator. He enriched his experience as an educator by working as an exteranal associate at the Department of Applied Computing at FER. Alongside his business engagement, he continues to develop his scientific career. His main fields of interest are data warehousing, business intelligence, data lakes, moving sensor data and IPTV. He is the author of eight articles published in scientific journals and conference proceedings.