Unidata — A Modern Master Data Management Platform

Sergey Kuznetsov[1], Alexey Tsyryulnikov[1], Vlad Kamensky[1], Ruslan Trachuk[1], Mikhail Mikhailov[1], Sergey Murskiy[1], Dmitrij Koznov[2] and George Chernishev[1,2]

[1] *Unidata*
[2] *Saint-Petersburg State University*

Abstract
Organizations need to ensure the quality of data that is used for analytics and to maintain its consistency across multiple analytical and operational systems. Master data is a term that refers to domain-specific data concerning business objects, crucial for organization operation, e.g., contracts, suppliers, employees and so on. Usually, such source data is scattered around different applications across the organization and is of varying quality. Master Data Management (MDM) is a set of practices, information management methods, and data management tools intended for producing accurate, consistent, and complete master data. At the same time, data management tools play a vital role in setting up and supporting MDM-related processes.

In this paper, we describe the Unidata platform: a toolkit for constructing MDM solutions. Its modular architecture allows to construct solutions tailored for a specific domain and case requirements.

We start with a short introduction to MDM, discussing its aims, user-facing benefits, and approaches to working with data. Then, we describe the architecture of the Unidata platform and present the data storage approach and query processing algorithms. Finally, we discuss use-cases and put forward our thoughts regarding the future of MDM systems.

Keywords
Master Data Management, Data Platform, Data Cleaning, Information Integration, Golden Record, Registry

1. Introduction

Contemporary companies and public institutions manage huge volumes of data, which is now a strategic asset [1]. Data became an enabler of organization business models and value propositions [2]. At the same time, each organization usually possesses a complex data ecosystem, which makes it hard to make use of information contained in it. For example, a recent study [3] concerning local governmental organizations in Denmark showed that one of the significant obstacles to using this data is the lack of its overview. Thus, if providing even an overview is hard, then efficient use will require much more effort which will include ensuring data quality (duplicate detection and data conflict resolution), setting up ETL pipelines, providing proper metadata management, tracking data lineage and so on.

Master Data Management (MDM) solutions aim to address these technical issues by consolidating available information while interacting with existing systems of organization in a minimally intrusive way. As a research discipline, MDM is an actively developing area that concerns all aspects of enterprise data management practices. Its practical counterpart is closely monitored by Gartner, which releases yearly reviews describing established and promising products. The recent publication of the 2nd edition of the fundamental reference [4] describing all aspects of MDM is also worth noting.

Large organizations that are principal customers of MDM vendors have many individual characteristics, and, consequently, individual pressing tasks. Therefore, by implementing MDM they aim to fulfill these tasks, thus adhering to the organization pull strategy [5]. The other strategy — technology push — is essentially large-scale adoption of a new technology based on the belief in its usefulness instead of focusing on particular tasks at hand. The organization pull approach significantly reduces the cost of MDM adoption by narrowing its application area. Moreover, its iterative nature enables progressive build-up of MDM functionality via step-by-step incorporation of different organization business areas. However, implementing MDM in the organization pull manner requires flexibility of the MDM toolkit used to build the solution, and major MDM toolkits fail to provide it since initially they were built as monolithic systems.

Usually, the structure of large organizations is the result of lengthy historical processes, which may include

Published in the Workshop Proceedings of the EDBT/ICDT 2022 Joint Conference (March 29-April 1, 2022), Edinburgh, UK
sergey.kuznetsov@unidata-platform.ru (S. Kuznetsov); alexey.tsyryulnikov@unidata-platform.ru (A. Tsyryulnikov); vlad.kamensky@unidata-platform.ru (V. Kamensky); ruslan.trachuk@unidata-platform.ru (R. Trachuk); mikhail.mikhailov@unidata-platform.ru (M. Mikhailov); sergey.murskiy@unidata-platform.ru (S. Murskiy); d.koznov@spbu.ru (D. Koznov); g.chernyshev@spbu.ru (G. Chernishev)
https://www.math.spbu.ru/user/chernishev/ (G. Chernishev)
0000-0001-7509-7888 (S. Kuznetsov); 0000-0001-6752-6742 (A. Tsyryulnikov); 0000-0002-9907-8945 (R. Trachuk); 0000-0001-6202-9034 (S. Murskiy); 0000-0003-2632-3193 (D. Koznov); 0000-0002-4265-9642 (G. Chernishev)

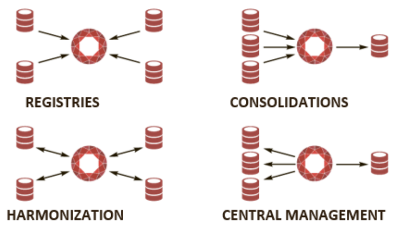

Figure 1: Approaches to handling data using an MDM solution

several acquisitions and mergers of other companies or organizations. Another important feature of such organizations is a unique software ecosystem comprised of core technologies from various vendors (e.g. SAP, IBM, Microsoft). Next, specific priorities and issues of the organization should be taken into account as well. Finally, there are various external constraints which an organization has to follow and which define the structure of the business processes. For example, organizations in different countries may have different rules, e.g. either a declaratory or an authorization system.

Next, the need to fulfill specific tasks of the organization leads to a specific subset of the MDM functionality of the solution being "clipped out" from an abstract general-purpose MDM toolkit. It may be due to tuning to organization software ecosystem or implementing specific functionality. Such functionality usually concerns areas that border on MDM or that may be completely external to MDM. Obviously, organizations do not want to have functionality and code that they do not need for the task at hand. Finally, there is a trend of implementing various enterprise applications in the cloud in order to lower expenses.

Multi-domain MDM, i.e. MDM covering several key business entity areas (products, vendors, employees and so on), is the state-of-the-art [6] approach. Aside from the aforementioned generic domains there may be a number of business-specific ones, such as patients in case of a hospital or an inventory of land plots managed by forestry. Furthermore, each customer may have its own set of domains, with its specifics. In order to support combinations of different domains, an MDM platform should be able to describe them inside itself and be sufficiently independent from any particular one.

In this paper we describe the Unidata platform — a core technology of an industrial product line [7] that was used to produce dozens of target MDM solutions for large organizations. These solutions are focused on the specific needs of organizations such as specialized business segments, heterogeneous IT infrastructure, and particular business tasks. The platform has received an honourable mention by Gartner in "Magic Quadrant for Master Data Management Solutions 2021" [8].

The architecture of the Unidata platform aims to address the above-mentioned concerns. Its core ideas are as follows.

1. **Tune-ability**. An MDM platform should be adaptable to various software ecosystems, take into account specific priorities and issues of the concrete organization, and comply with external constraints. This will ensure that target MDM solutions meet the organization needs.
2. **Component-based architecture**. In order to provide the required flexibility, the MDM platform architecture has to follow the component-based principle. The platform should consist of modules — i.e., building blocks which can be freely combined together in order to create efficient and specialized MDM solutions.
3. **Metamodeling**. A metamodel describing models of individual areas can be used to handle combinations of different domains. This will provide the product with the required extensionability.

Building an MDM platform upon these principles makes it possible to achieve synergy between individual MDM instruments such as data quality, data provenance, metadata management, and so on. That is, building a solution from tools that are intended to work together from the start (as opposed to integrating a set of different tools) will result in shorter development times and solutions of better quality. Furthermore, it will be possible to deliver different versions of solutions with different functionality (e.g. standard vs enterprise edition) and prices. The overall idea of this approach is presented in Figure 2. The core set of modules has been released as an open-source version[1] of the platform. It is published under the GNU GPLv3 license.

Overall, the contributions of this paper are:

1. A description of the Unidata platform architecture.
2. A discussion of data storage and processing algorithms.
3. A list of use-cases and our vision of the future of MDM.

The structure of this paper is as follows. We start with the description of the background in Section 2. In it, we provide basic definitions and describe user roles. Next, in Section 3 we discuss several use-cases in order to illustrate benefits that can be obtained by using such systems. All presented use cases are real and describe actual MDM deployments that were driven by the organization pull strategy. Section 4 contains the description of the platform architecture and its modules. The way data is stored

[1]https://gitlab.com/unidata-community-group/unidata-platform-deploy

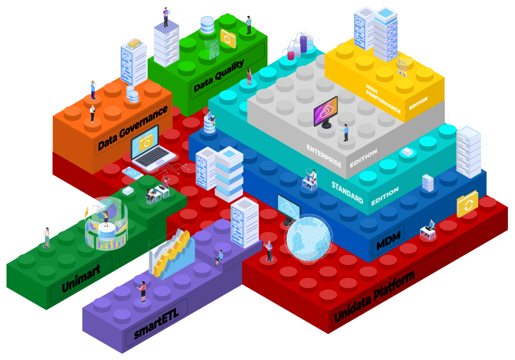

Figure 2: Modular structure of the Unidata platform

and how queries are processed is discussed in Section 5. Next, present our view on the future of MDM systems and describe related work in Sections 6 and 7. Finally, we conclude this paper with Section 8.

2. Background

2.1. Master Data Management

According to David Loshin [9], Master Data "are those core business objects used in the different applications across the organization, along with their associated metadata, attributes, definitions, roles, connections, and taxonomies". Examples are product data, customer data, supplier data, location data, party data, reference data, asset data, employee data, ledger data, and vendor data.

In turn, Master Data Management is "a set of practices, information management methods, and data management tools to implement the policies, procedures, services, and infrastructure to support the capture, integration, and subsequent shared use of accurate, timely, consistent, and complete master data".

The purpose of an MDM platform is to obtain data from source information system(s), then process it to ensure data quality by, for example, performing data deduplication, filling in missing values, removing outdated information, etc. Eventually, it must obtain a golden record for each item — an error-free version conforming to the defined quality criteria.

How exactly data is processed is defined by the MDM implementation architecture/style [6, 10] (sometimes called the data hub architecture). Gartner identifies four approaches, presented in Figure 1).

1. Registry. The hub does not contain the data itself, instead storing only the corresponding references (indexes). This approach is relevant for data that cannot be copied or "moved" for various (including legal) reasons.
2. Consolidation. Data is uploaded into the common repository on a regular basis, appropriately processed, and then the hub itself provides data consuming systems with access to this data. However, new data is supplied by live data source systems, i.e., new data is uploaded to the hub on a regular basis.
3. Centralization. This architecture is very similar to the previous one, but here the hub takes over data upload as well: i.e., data is uploaded once, and then all changes are performed on the hub itself, thus turning all systems that initially were data sources into data consumers.
4. Coexistence. This architecture implements a combination of the Consolidation and Centralization for different master data of an organization. Additionally, if some data fragments are not "movable", they can be handled using the Registry.

All of them are supported by the Unidata platform.

2.2. Basic definitions

Golden (master) record. According to [6], one of the core goals of MDM systems is to create and maintain a single version of the truth for an entity. The information which constitutes it is stored in multiple sources and thus, it should be assembled. All information concerning a particular entity is called the golden record.

Data models and Metamodel. In order to support a particular domain, the objects which the platform will work with must be defined. A Metamodel consists of the description of the data itself (data schema) and related procedures. For example, it is possible to add a supplementary metamodel of data quality for a particular domain, e.g. specific duplicate detection procedures.

In other words, a metamodel specifies how individual data models will be created and processed.

A **registry** is a collection of records that are related to some entity, such as a person, an organization, etc. This information comes from many different source information systems. A registry has a schema which consists of a list of its attributes and nested entities. Similarly to tables, registries can have references to other registries, e.g. suppliers and items: each supplier can have references to items which it sells.

A **lookup** table is a referential table that contains data which is rarely changed, but at the same time frequently used. For example, lookup tables may be created to list countries, timezones, currencies, and so on. Similarly to registries, lookup tables consist of attributes.

System of a Record (SOR). During the golden record assembly process, the same attribute may turn out to be

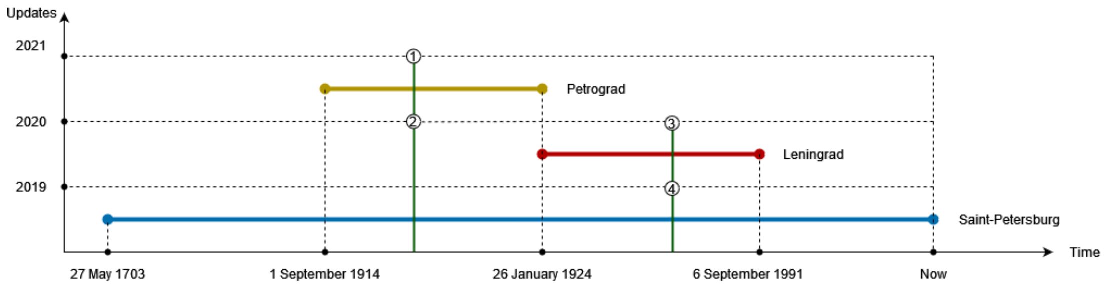

Figure 3: Representation of the Saint-Petersburg name validity periods

stored in several sources, having different values. In this case, either a system of a record or a conflict resolution process should be set up. SOR is a primary data source which contains the "true" value.

Validity Period — is an interval in which the data describing an entity is valid. Each golden record may have several validity periods which should be taken into account while querying the data. Moreover, there are two temporal dimensions: time of an event and time of addition of this new version of information into the system. This leads to a need for a special scheme for managing this information.

Consider the example presented in Figure 3 which describes the history of Saint Petersburg's name changes. The Y-axis denotes update times, while the X-axis shows the validity periods. In this example, we assume that before 2019 the system contained only the basic version of name information: from 1703 until the present time (2019), the city was called Saint Petersburg. Then, somewhere between 2019 and 2020, the knowledge of the Leningrad name was added into the system, and later, somewhere before 2021, a similar update was done for Petrograd.

A user may pose queries like "what was the name of the city in 1921". There are two possible answers: up until 2020 (point $\textcircled{1}$) the system would have returned "Saint Petersburg", which was the correct answer at the time. Currently, (point $\textcircled{2}$) it should output "Petrograd".

Thus, the golden record should be constructed on-the-fly, taking into account updates in two temporal dimensions which is done by looking into the origin history (which is discussed later).

2.3. Humans in the MDM system

From a high-level perspective, an MDM system produces a new role for a human user — a Data Steward. According to [9], a data steward is a person responsible for collecting, collating, and evaluating issues and problems with data and the data life cycle. Their duty includes managing standard business definitions and metadata. Finally, a data steward is the person who is accountable for the quality of the data. However, in practice, a data steward is not necessarily a single person, but rather, a group of several people. Each of them may be responsible for a different key business area or even a part of it. The Unidata platform supports two types of users that implement data stewardship:

- Administrator manages the platform in general. Their duties include data model administration, classifier[2] administration, managing rules of duplicate detection and so on.
- Data operator manages records (e.g. database population), registries and lookup tables, and participates in related business processes.

Consequently, there are two types of interfaces, for each type of user. Finally, Unidata platform supports role-based access model, where each user may be assigned rights to perform operations (e.g. CRUD) with various objects.

3. Use Cases

All of the presented use-cases are real projects that were completed using the Unidata platform, driven by the organization pull strategy. They clearly illustrate the concrete goals of organizations, as well as the benefits the platform has provided to the final users.

They also demonstrate that each particular deployment needs a different set of data governance tools, i.e. they highlight the importance of **component-based architecture**, which we have discussed in the Introduction.

3.1. Case 1: managing logistical resources

The first case is a system for managing logistical resources of a large company in the energy sector. The

[2]In this paper, "classifier" refers mainly to a product class hierarchy, such as the Global Product Classification [11].

covered domains included raw materials, equipment, and replacement parts. The purpose of this system was to:

- Provide high-quality data for business processes that cover equipment maintenance and repairs, inventory management.
- Consolidate available information using data standardization and unification.

Additionally, this project succeeded at automating complex company regulations that involved more than ten different divisions. Furthermore, a classifier of logistical resources was deployed.

3.2. Case 2: product catalogue for a telecommunication company

The next case is a product catalogue development for a large telecommunication company. The goals of the project were to consolidate:

- information regarding product offers for different customer segments,
- information related to service availability,
- financial information from the billing system and accounting records.

As the result, a product hierarchy (a product tree) that contains various product details (including financial ones) was constructed. It is now utilized by the sales department and financial officers.

3.3. Case 3: data consolidation for a transport company

This project was dedicated to consolidation of items and services purchased by a large transport company. The goal was to unify information contained in several different product classifiers and to produce a list of services offered by contractors.

The stakeholder of this MDM solution was the procurement department of the company. After the deployment, items and services that had different prices were identified and analyzed. This resulted in creation of monetary metrics, i.e., calculated total savings on purchases. The system made it possible to perform automatic purchases with the minimal available price.

This data consolidation project relied on the publish/subscribe model.

3.4. Case 4: data enrichment for a fashion vendor

This project concerned an MDM system for a company that sells fashion products. The system needed to perform client base segmentation and sales support in the premium segment.

The aim of the system was to find those clients in the client database that have popular social media accounts and a lot of followers. The company wanted to improve their loyalty by offering additional discounts and performing various other actions in order to obtain more customers from their follower bases. The core data governance tools that were used were data enrichment and consolidation.

3.5. Case 5: smart personal account

The goal of this project was to create a smart personal account of a city resident. It was necessary to integrate it with various federal and regional information systems. The reason for this was to enable exporting relevant information concerning vehicles, real estate, bank accounts and so on. The primary focus of this project was access control, security, and ensuring real-time as well as publish/subscribe model master data acquisition.

3.6. Case 6: energy and heavy industry company

This project was developed for a multi-sector international company focused on energy and heavy industry areas. Since this company has hundreds of thousands of clients from all over the world, the procedure of adding a new client was very complicated. Before applying our MDM solution, it took 21 days on average. However, it has shortened to only eight days after. The solution automatized various checks, finding final beneficiaries in corporate hierarchies, and centralized information input.

This project mainly concerned data inventory, data quality (duplicate search), and implementing real-time access to master data in order to speed up a particular business process.

4. Architecture

Now, let us turn to the architecture of the Unidata platform. It follows the component-based principle, which means its building blocks (modules) can be freely combined with each other in order to obtain a solution with the desired set of features.

4.1. Preliminaries

A **module** is a self-sufficient set of functionality that is intended for solving a particular problem. Each module contains a number of services that cover parts of this functionality. For example, the Meta module which encompasses all metadata-related activities contains services that cover managing lookup tables, registries, units of measurement, enumerations, etc.

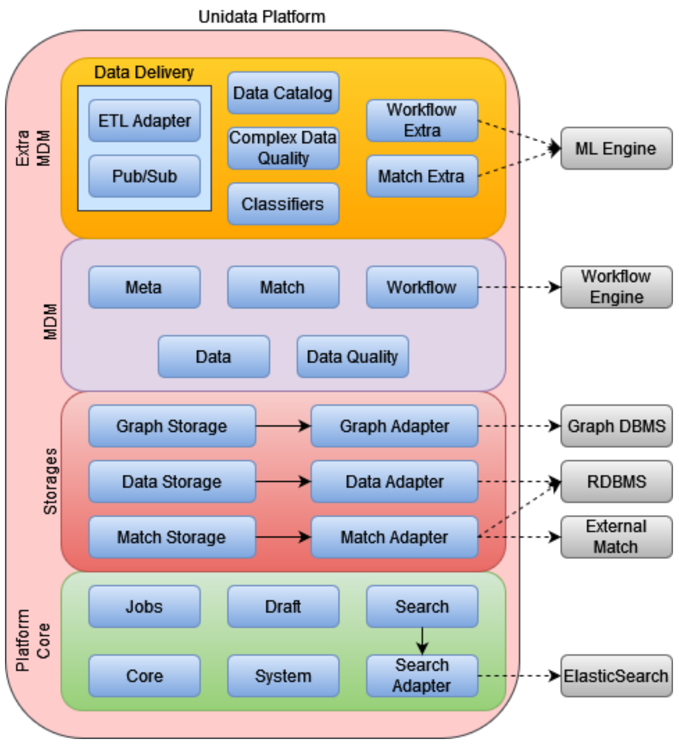

Figure 4: Architecture of UniData Platform

Modules have rules of creation (a contract), behavior, and they can interact with other modules via being a part of a pipeline.

In a broad sense, a **Pipeline** is a sequence of operations which are performed either on data or on the model. Pipelines implement dataflows, which consist of service calls and may contain utility nodes such as branching, parallelism (applying an operation to each record inside the batch), calling another pipeline, and so on.

Each module contains services that work with the data and services that interact with and modify the model. Therefore, pipelines may modify not only the data, but the model itself too.

Thus, modules and pipelines implement tuneability and composition aspects discussed in the Introduction section.

4.2. Architectural Overview

The overall architecture of the Unidata platform is presented in Figure 4. It consists of four main components:

1. **Platform Core** contains basic modules that implement core services of the platform and minimally depend on other modules. These services are: system boot, job batches, and data types. Modules of this component are rarely modified and are never adapted for a particular deployment or domain.
2. **Storages** concern everything related to data stores that are employed by the platform. These modules make it possible to abstract functionality required by the platform from the specific DBMS and to restrict and simplify it (as not all DBMS functionality is required by an MDM solution).
3. **MDM** includes all modules that implement basic MDM functionality, such as metadata management, rules for computing master record alternatives, data quality management, duplicate detection, and business process implementation. This component is responsible for integrating and synchronizing all necessary data stores that reside on the previous level.
4. **Extra MDM** implements advanced MDM functionality. It contains modules that are either advanced variants of some existing module from the MDM component, or modules that provide functionality usually implemented “outside” of MDM. An example of the first case is the following: Match Extra is a sophisticated machine learning-based inexact match module and the

MDM match module is a straightforward exact one. The second case is illustrated by the data delivery set. Usually, in MDM systems the ETL is implemented separately as a standalone application. In our case, it is possible to have it inside the system. The same idea is employed with the Pub/Sub module, which make it possible to implement sophisticated patterns of sending records to various consumers. This is an advanced component which is not present in the open-source version.

These components are organized in a hierarchical way, which means that:

- components residing in the bottom levels of the figure are more low-level, they contain basic features, essential for implementing high-level functionality, and,
- components interact with each other in hierarchical way, i.e. their interactions rarely "jump" over the immediate neighbor.

4.3. Entities, modules, and their relationships

In this section, we will overview core entities that the system works with, as well as important modules and their functionality relevant to the entities.

System. All modules that constitute the platform share a single interface that contains methods of initialization, configuration, launch, and verification. The system module orchestrates system boot process and ensures that all modules have everything necessary for trouble-free launch and operation.

Data Types. An attribute is a basic entity representing some key aspect of a stored object, similar to attributes in RDBMS tables. In Unidata, Registries, lookup tables and references may have attributes.

There are three attribute types in the system:

1. **Simple attribute** is a basic data type which describes some entity. Its type can be: string, numeric, boolean, file, date and time, reference, and enum. Enum is a domain-specific enumeration which describes mutually exclusive states of a real-life entity, e.g. a subject which can be either a legal person, a natural person, or a self-employed person.
2. **Array attribute** is used to represent a series of similar entities, such as property owners.
3. **Complex attribute** is used to represent nested tree-like structures. It can contain simple attributes, arrays, and other complex attributes.

The Unidata platform also supports references, which can be of the following types:

1. Reference — a reference which ensures that for each individual record there can be only one referenced object per each validity period.
2. Many-to-many — a classic many-to-many reference.
3. Contains — a reference which is created by user pointing to an entity with all attributes filled in. This type is used to define entities which do not exist without its parent entity.

The platform lets the user browse both direct and backward references (those that point to a specific record). It also provides rich search capabilities that allow the user to query attributes of references.

The **Core** module contains interfaces and abstract implementations for all data types that the platform uses. It also implements **metamodel** support, the need for which was discussed in the Introduction. Finally, this module supports additional services which may be used by other modules such as roles, logging, license verification and so on.

The next important entity is a **draft**. Drafts are an essential part of MDM since master data needs to be synchronized over data producers (sources) and data consumers. Such synchronization frequently results in the creation of temporary intermediates. These need to pass various reconciliation procedures and be agreed upon in order to become fair copies. Drafts may also emerge as a result of conflicts that arise during record consolidation process, after various data quality procedures, etc.

Drafts may concern individual items or the model. Drafts can have revisions: all of them are stored in the database in a serialized form.

Therefore, drafts need to have general support: operations such as creation, publication (transforming into fair copy), and merge must be implemented. **Draft** is a module that enables all kinds of drafts in the platform.

Logging capabilities. The Unidata platform supports extensive and configurable logging capabilities. It is possible to record user actions such as record upsert, user login or logout, etc. It is also possible to select the logging level. Coarse-grained logging logs errors only, while fine-grained enables the logging of search and browsing.

Storages and Adapters. For its operation, Unidata needs a number of storages, e.g. data storage, graph storage, match storage, and so on. In order to achieve flexibility and independence from a particular database vendor, a collection of adapters was implemented. For example, the platform can use either Neo4J or OrientDB for graph storage.

Match storage is a module that concerns data representation for performing data deduplication. There is a separate representation for both records and clusters that differ from the original data by, for example, omitted fields that do not participate in the matching. For the deduplication itself, matching engines such as Senzing, Elasticsearch or RDBMS can be used.

Match and Match Extra. While match storage implements basic matching functionality, this module implements deduplication in MDM entities: in the data itself, drafts, business processes, etc. The Match Storage module does not possess any specifics regarding these entities and therefore separate modules are needed. These modules also let the user define deduplication procedures and manage them. The Match module operates with rules, while Match Extra employs various machine learning approaches.

Workflow and Workflow Extra. In MDM, it is frequently necessary to implement various business processes that may involve various officials and span multiple departments (e.g. resolving a surname conflict in various personal documents). To run such workflows, a subset of the BPMN 2.0 standard that includes events, activities, and gateways is supported. For this, integration with several third-party engines such as Activiti BPMN and Camunda BPMN is implemented.

Workflow Extra extends basic capability by enabling the implementation of various scenarios involving machine learning approaches. It also provides the ability to perform SLA enforcement for data operations.

Meta is a module that is responsible for metamodel (schema) management such as creating and editing entities (attributes, references, etc). It provides a GUI that allows users of the MDM solution (data stewards) to work with the metamodel while automatically generating the corresponding API for data access.

Data Quality and Complex Data Quality. These modules are responsible for ensuring data quality — checking data for errors and performing data enrichment. While the Data Quality component works with a single record, Complex Data Quality may involve several records (e.g. checking aggregate values).

The core instruments are the enrichment and validation rules. The enrichment rule enables using attributes of other registries in order to generate new attributes of the target registry. The validation rule does not generate any data, instead, it generates an error if it is violated by input. Each rule can have a number of ports which may be either incoming or outgoing. Rules take input from the incoming ports and produce results into the outgoing ones.

In the UI, a user can construct rules using a special UPath language (an XPath derivative) and a number of pre-defined functions such as string manipulation functions (case conversion, concatenation), a boolean function and so on. It is possible to upload custom rules that are implemented in Java or Python.

Finally, rules can be grouped into sets and applied on a per-set basis.

Classifiers. This module employs classifiers for master data. Classifiers are tree-like data structures in which each node describes some entity and may contain various attributes (e.g. see [11]). They are frequently used in MDM domains, for example, to describe a hierarchy of product types. Unidata platform enables interactions of master data with such classifiers (e.g. arranging arriving data according to a classifier) and supports various operations on classifiers themselves with versioning.

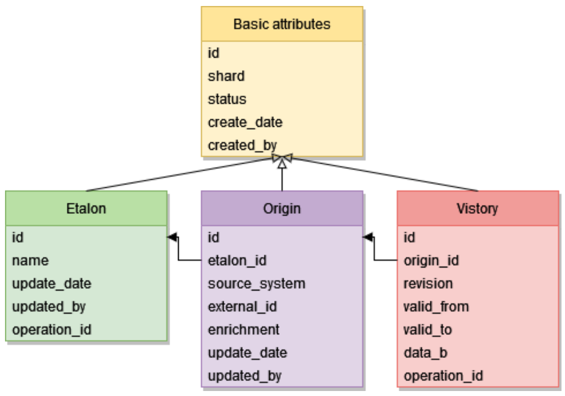

Figure 5: Tables used for data storage in the Unidata platform

Data Catalog implements a comprehensive body of knowledge concerning all data of the organization. It includes provenance, current storage location, its domain, use, relation to other data and so on. This module is of critical importance since an organization may have hundreds of source information systems and manual tracking of such information would be impossible.

5. Data Storage and Processing

5.1. Requirements

The specifics of the platform's application scope lead to the following requirements imposed on storage and processing of the data.

1. Tombstone deletes. Information should be deleted only when it is absolutely necessary. Real deletions should be performed only by an administrator while regular users should not actually delete data. Instead, if a regular user attempts to perform a deletion, the data should be marked as deleted and the system should take this into account.
2. Versioning support should be pervasive. Users and administrators should be allowed to reconstruct previous versions of any object. At the same time, querying using validity periods, discussed earlier in Section 2.2 should be supported.

3. Provenance (traceability) should be provided for any operation. For example, if a bulk-loading operation inserted records into a database, it should be possible to reverse it by removing newly-inserted records while keeping the rest.

These points should be fulfilled for all data handled by the system, even for the manually entered.

5.2. Storage

In order to meet these requirements, an approach based on the following three logical tables was used. Each table represents an entity:

- Etalon is the metadata of the golden record itself.
- Origin is the metadata related to system from which the record originates.
- Vistory (version history) is the validity period of origin, which in turn may have revisions.

Tables describing these entities share the following attributes.

1. Id — an unique identifier of an object.
2. Shard — an identifier of the shard where the record resides. Each of these logical tables may be physically represented by a collection of horizontal partitions (shards).
3. Status — may be ACTIVE, INACTIVE, or MERGED. These are used to describe the status of the record and to support tombstone deletes. Thus, INACTIVE means that the record was deleted, while ACTIVE indicates that it is valid. The MERGED status indicates records that were used in the duplicate resolution process and no longer contain valid information.
4. Create_date and Created_by — when and by whom this object was created.

The relations between these tables are shown in Figure 5. The links with empty arrowheads denote “shared” attributes and full arrowheads show PK-FK relationship.

The etalon table additionally stores:

1. Name of the registry or lookup table it refers to.
2. Update_date and update_by attributes, which contain the date and the last user who updated this data, respectively.
3. Operation_id attribute which contains the identifier of the operation during which this record was created. It is needed to support the provenance requirement.

Apart from the basic set of attributes, the origins table has the following additional ones:

1. Etalon_id — an identifier of etalon which this origin belongs to.
2. Source_system — an identifier of the system where this record came from.
3. External_id — an identifier of the record in the system where this record came from.
4. Enrichment — a boolean flag which shows whether this origin is a result of the enrichment rule.
5. Similarly to etalon, attributes update_date and update_by that bear the same semantics but pertaining to this particular origin.

Finally, the vistory table contains the data itself. Its important attributes (apart from the basic ones) are the following:

1. Origin_id — a reference to origin whose part of vistory is contained in this record.
2. Revision — a version number, which is needed to ensure versioning, described in Section 5.1.
3. ValidFrom and validTo attributes — a validity period of the vistory entry.
4. Data_b — serialized data in the XML format.
5. Operation_id attribute, similar to the one in the Etalon table. It can be used to, for example, find and cancel a modification action. However, in this case, cancelling will affect only this particular update, while in the etalon case it will cancel the creation of the whole record.

Note that the vistory table contains no attributes concerning updates, since for each update a new record is formed. Next, setting the status attribute INACTIVE in case of the vistory table makes it possible to mark some validity period as “deleted”.

To illustrate the idea, in Table 1 we provide a vistory table fragment for the city name example described in Figure 3. There are three records and all of them belong to a single origin. Therefore, there are no data conflicts possible and the process of calculating the etalon data is rather straightforward. First, it is necessary to perform a cross-product of all time periods, and then to select the row that fits the necessary combination of creation and queried dates. The result of the period cross-product for the considered example is shown in Table 2. Thus, it is possible to find an answer for points ①–④.

Finally, there are complex rules of attribute interaction inside the etalon $\longrightarrow$ origin $\longrightarrow$ vistory hierarchy. For example, if adding a new revision of a particular origin leads to a recalculation of the etalon, then its update time is recalculated too. Next, status updates are propagated in a bottom-up fashion: e.g. if INACTIVE is set for a vistory entry, then its etalon will have to be recalculated and may have to be set INACTIVE too. However, if etalon is set INACTIVE, then there is no need to set all vistory

Table 1
Vistory representation of the city name example from Figure 3

city_name	rev	validFrom	validTo	createDate
Saint-Petersburg	1	27.05.1703	31.12.9999	01.01.2018
Leningrad	2	26.01.1924	6.09.1991	01.06.2019
Petrograd	3	1.09.1914	26.01.1924	01.06.2020

Table 2
Validity periods calculation

city_name	rev	validFrom	validTo	createDate
Saint-Petersburg	1	27.05.1703	1.09.1914	01.01.2018
Petrograd	2	1.09.1914	26.01.1924	01.06.2020
Leningrad	3	26.01.1924	6.09.1991	01.06.2019
Saint-Petersburg	1	6.09.1991	31.12.9999	01.01.2018

entries INACTIVE since they will never be reachable for queries.

The same approach is used to represent not only records, but other entities as well, e.g. references and classification results. It is possible to create a golden record for a reference. Consider a case where a reference has two origin systems, and in the first system, there is a set_1 of values and in the second a set_2. Using the BVR or BVT algorithm, it is possible to create a golden record, by, for example, intersecting them.

5.3. Query Processing: BVR and BVT algorithms

Each vistory entry has a date of creation and a validity period, which are used to construct golden records, as was demonstrated in the previous section. However, what if there are two origin systems which have the same attribute and there is a data conflict, i.e. each system reports different values? In other words, how is a SOR selected for this attribute?

For this, two special algorithms — the BVR (Best Value Record) and BVT (Best Value of the Truth) — were devised. The BVR algorithm is used to construct a golden record by resolving data conflicts for all attributes of an etalon using a set of thresholds, one for each origin system. The general idea of this algorithm is to pick values from source systems that have higher weights.

More formally, the BVR algorithm is as follows.

1. For each origin obtain its latest version, except the ones that have the MERGED status.
2. For each source_system select the latest version according to its creation_date.
3. The golden record will be created out of the record that pertains to the source_system with the maximum weight.

The BVT algorithm is used to construct the golden record when system administrator wishes to form it on a per-attribute basis.

The algorithm itself is as follows:

1. Similarly to the BVR, obtain the latest version for each origin, except origins that have the MERGED status.
2. For each attribute sort obtained versions according to the weights of sources and update_date
3. Compute values of each attribute by iterating over versions obtained on the previous step:
   a) if the value of the attribute is not null, then use it for etalon construction.
   b) otherwise, proceed to the next iteration.

Note that BVT algorithm is meant to be more robust to null values and therefore handles them differently.

To illustrate both algorithms, let us consider the example presented in Table 3. In this table, we denote entities selected on each step in bold. The first seven rows contain the data itself. Note that for presentation purposes we have joined all three tables that contain it.

Our first step (regardless of the selected algorithm) is to select most recent vistories for each origin, which is done inside the DBMS. Rows 15–16 contain the answer.

On the second step it is necessary to compute validity periods, which is done in Java code (all consequent computations are performed in Java code, too). Applying cross-product to validFrom and validTo attributes, we obtain three periods: (1989–2000), (2000–2005), (2005–9999). Rows 20–23 contain our data partitioned by validity periods.

One can note that there is a data conflict, namely, rows 20 and 23 concern the same period and contain different values. Periods of rows 21 and 22 do not have alternatives and therefore may be used as is.

In order to resolve the conflict, the BVR algorithm will require weights of source systems. Suppose that they are as follows: source1 = 50, source2 = 100. In this case, the record on row 32 will be selected with all its attributes.

The BVT algorithm will additionally require a set of attribute weights. In this example, we have weights for a single attribute — year_of_birth, which are as follows: source1 = 100, source2 = 50.

These attribute weights are used to override source weights that act over the whole record. Therefore, the year_of_birth attribute will be set as 1991, while name attribute will be set as John.

The BVT algorithm additionally follows the "null is not a value" rule. It never picks null value, even if it has to according to an attribute rule. This is why we select Saint-Petersburg in the city attribute. At the same time,

Table 3
BVT and BVR illustration

1	Initial data									
2	etalon_d	origin_d	source_system	name	city	year_of_birth	validFrom	validTo	rev	create_date
3	etalon1	origin1	source1	John	Moscow	1991	2000	9999	1	9.11.2021
4	etalon1	origin1	source1	John	Saint-Petersburg	1991	2000	9999	2	10.11.2021
5	etalon1	origin2	source2	John		1992	1989	2005	1	8.11.2021
6	etalon1	origin2	source2	John		1993	1989	2005	2	11.11.2021
7	1. Selecting actual vistories									
8	etalon_d	origin_d	source_system	name	city	year_of_birth	validFrom	validTo	rev	create_date
9	etalon1	origin1	source1	John	Moscow	1991	2000	9999	1	9.11.2021
10	**etalon1**	**origin1**	**source1**	**John**	**Saint-Petersburg**	**1991**	**2000**	**9999**	**2**	**10.11.2021**
11	etalon1	origin2	source2	John		1992	1989	2005	1	8.11.2021
12	**etalon1**	**origin2**	**source2**	**John**		**1993**	**1989**	**2005**	**2**	**11.11.2021**
13	Result									
14	etalon_d	origin_d	source_system	name	city	year_of_birth	validFrom	validTo	rev	create_date
15	etalon1	origin1	source1	John	Saint-Petersburg	1991	2000	9999	2	10.11.2021
16	etalon1	origin2	source2	John		1993	1989	2005	2	11.11.2021
17	2. Calculation of the validity periods.									
18	Using cross-product to obtain the following periods: (1989-2000), (2000-2005), (2005-9999). Result in the vistory form:									
19	etalon_d	origin_d	source_system	name	city	year_of_birth	validFrom	validTo	rev	create_date
20	etalon1	origin1	source1	John	Saint-Petersburg	1991	2000	2005	2	10.11.2021
21	etalon1	origin1	source1	John	Saint-Petersburg	1991	2005	9999	2	10.11.2021
22	etalon1	origin2	source2	John		1993	1989	2000	2	11.11.2021
23	etalon1	origin2	source2	John		1993	2000	2005	2	11.11.2021
24	3. Calculation of the golden record.									
25	Records participating in consolidation:									
26	etalon_d	origin_d	source_system	name	city	year_of_birth	validFrom	validTo	rev	create_date
27	etalon1	origin1	source1	John	Saint-Petersburg	1991	2000	2005	2	10.11.2021
28	etalon1	origin2	source2	John		1993	2000	2005	2	11.11.2021
29	Example 1: BVR. BVR settings: source1 = 50, source2 = 100.									
30	etalon_d	origin_d	source_system	name	city	year_of_birth	validFrom	validTo	rev	create_date
31	etalon1	origin1	source1	John	Saint-Petersburg	1991	2000	2005	2	10.11.2021
32	**etalon1**	**origin2**	**source2**	**John**		**1993**	**2000**	**2005**	**2**	**11.11.2021**
33	Example 2: BVT. BVT Settings: year_of_birth: source1 = 100, source2 = 50									
34	etalon_d	origin_d	source_system	name	city	year_of_birth	validFrom	validTo	rev	create_date
35	etalon1	origin1	source1	John	**Saint-Petersburg**	**1991**	2000	2005	2	10.11.2021
36	**etalon1**	**origin2**	**source2**	**John**		1993	**2000**	**2005**	**2**	**11.11.2021**

this rule is absent in the BVR case, as was demonstrated in row 31.

Note that we have computed data for the period that had data conflicts, but the considered golden record spans multiple periods. The overall result for both algorithms that includes data for all periods is shown in Table 4.

6. Beyond MDM

Despite recent significant technological advances, human approaches to handling information have not really changed. This is also true in case of MDM systems, in which interaction scenarios have stayed the same. Users still have to think about data layouts and procedures, e.g., define registries and referential tables, set up data pipelines and so on. In other words, people solve problems in an imperative way, specifying everything that needs to be done in order to obtain answers. This approach requires a lot of effort, which often comes in the form of duplicated work since processes share a large degree of similarity in many organizations.

At the same time, humans think in terms of problems such as: "Why did sales drop in the last quarter?" or "How did the introduction of the new discount system impact profits?". A declarative approach would suit these problems better, and thus there is a need for it.

There are two possible ways to achieve it — exhaustive standardization and machine learning. The former is very challenging since there are too many details that should be reflected inside the standards. Companies and public institutions exist all over the world and each has to conform to various local and international regulations. Standardizing them all will either require a coordinated

Table 4
BVT and BVR result comparison

BVR					
etalon_id	name	city	year_of_birth	validFrom	validTo
etalon1	John		1993	1989	2000
etalon1	John		1993	2000	2005
etalon1	John	Saint-Petersburg	1991	2005	9999
BVT					
etalon1	name	city	year_of_birth	validFrom	validTo
etalon1	John		1993	1989	2000
etalon1	John	Saint-Petersburg	1991	2000	2005
etalon1	John	Saint-Petersburg	1991	2005	9999

effort of multiple governmental entities or lead to immense labor costs required to pay the workers who will perform this standardization.

Machine learning, on the other hand, will incur smaller costs and will not depend on any external collaboration. While a full-fledged AI with a natural language interface is a very distant vision, machine learning has already been successfully adopted in individual components of MDM systems. For example, semantic column type detection [12, 13, 14], database schema matching [15], duplicate detection [16, 17, 18], and various types of table autocompletion [19, 20].

Another promising direction is digital storytelling [21, 22, 23] — automatically extracting and presenting facts contained in the data in a human-friendly way. Employing such techniques will lower the qualification requirements and open analytics to broader public.

There is also a novel class of so-called visual analytics [24, 25] systems that contain collaborative tools that allow users to employ various visualization primitives, machine learning models, and other objects. These are dragged onto a dashboard and connected to each other in order to construct pipelines. Thus, machine learning will be present not only inside such systems, but outside as well, i.e. allow users to build and use custom machine learning models inside their decision-making pipelines.

7. Related Work

The established MDM market vendors [8] such as IBM, SAP, Informatica and others offer a wide range of products for the creation of all types of MDM systems.

However, their toolkits 1) were largely started as monolithic products, 2) are heavily oriented towards vendors' infrastructure, 3) are frequently proprietary software which is not open-sourced. While the monolithic approach greatly simplifies architecture, it has a number of drawbacks, such as hindering extension-ability and thus making open-sourcing largely useless. Vendor orientation is not necessarily a bad thing, but the need to cope with a systems zoo requires modern MDM products to be flexible in terms of the used DBMS, search, BPMN implementation and so on. Not every customer is willing to add more dependencies, which may also require additional expenses.

However, the next generation of MDM toolkits such as Egeria[3], Fuyuko[4], AtroCore MDM[5] and many others offer open-sourced versions and actively attempt to implement a modular architecture.

The reason for this are changes in the MDM landscape and emerging requirements. The contemporary environment favors, if not requires modular and open products. Cloud-ready systems have become mainstream, and these properties are a must in ensuring extension-ability.

Finally, aiming for the organization pull strategy, one must prefer the latter approach, since such applications require increased flexibility. The Unidata platform aims for this niche and is therefore modular, open-source and extension-able.

8. Conclusion

In this paper we have presented the Unidata platform — a software product line intended for the creation of various MDM solutions. We have described its architecture, use-cases, data storage and query processing algorithms. Finally, we have shared our vision regarding the future of MDM systems.

Acknowledgments

We would like to thank Alexander Konstantinov and Roman Strekalovsky for their comments. We would also like to thank Anna Smirnova for her help with the preparation of the paper.

[3]https://github.com/odpi/egeria
[4]https://github.com/tmjeee/fuyuko
[5]https://github.com/atrocore/atrocore

References

[1] V. Khatri, C. V. Brown, Designing data governance, Commun. ACM 53 (2010) 148–152. URL: https://doi.org/10.1145/1629175.1629210. doi:10.1145/1629175.1629210.

[2] M. Jagals, E. Karger, F. Ahlemann, Already grown-up or still in puberty? a bibliometric review of 16 years of data governance research, Corporate Ownership & Control 19 (2021) 105–120.

[3] O. B. Nielsen, et al., Why governing data is difficult: Findings from danish local government, in: Smart Working, Living and Organising, Springer International Publishing, Cham, 2019, pp. 15–29.

[4] D. International, DAMA-DMBOK: Data Management Body of Knowledge (2nd Edition), Technics Publications, LLC, Denville, NJ, USA, 2017.

[5] R. W. Zmud, An Examination of 'Push-Pull' Theory Applied to Process Innovation in Knowledge Work, Management Science 30 (1984) 727–738. URL: https://www.jstor.org/stable/2631752, publisher: INFORMS.

[6] M. Allen, D. Cervo, Multi-Domain Master Data Management: Advanced MDM and Data Governance in Practice, 1st ed., Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2015.

[7] Software Product Lines. Carnegie Mellon Software Engineering Institute Web Site., https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=513819, 2022.

[8] Walker S., Parker S, Hawker M., Radhakrishnan D., Dayley A., Magic Quadrant for Master Data Management. Gartner. ID G00466922., https://www.gartner.com/en/documents/3995999/magic-quadrant-for-master-data-management-solutions, 27 January 2021.

[9] D. Loshin, Master Data Management, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2009.

[10] Andrew White. The Five Vectors of Complexity That Define Your MDM Strategy. ID: G00276267., https://www.gartner.com/en/documents/3038017/the-five-vectors-of-complexity-that-define-your-mdm-stra, 27 April 2015.

[11] Global Product Classification (GPC). GS1 Web Site., https://www.gs1.org/standards/gpc, 2022.

[12] M. Hulsebos, et al., Sherlock: A deep learning approach to semantic data type detection, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '19, 2019, p. 1500–1508.

[13] X. Deng, et al., Turl: Table understanding through representation learning, Proc. VLDB Endow. 14 (2020) 307–319.

[14] D. Zhang, et al., Sato: Contextual semantic type detection in tables, Proc. VLDB Endow. 13 (2020) 1835–1848.

[15] T. Sahay, A. Mehta, S. Jadon, Schema matching using machine learning, CoRR abs/1911.11543 (2019). arXiv:1911.11543.

[16] N. Barlaug, J. A. Gulla, Neural networks for entity matching: A survey, ACM Trans. Knowl. Discov. Data 15 (2021).

[17] W.-C. Tan, Deep data integration, in: Proceedings of the 2021 International Conference on Management of Data, SIGMOD/PODS '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 2.

[18] Y. Li, et al., Deep entity matching: Challenges and opportunities, J. Data and Information Quality 13 (2021).

[19] S. Zhang, K. Balog, Web table extraction, retrieval and augmentation, in: B. Piwowarski, M. Chevalier, É. Gaussier, Y. Maarek, J. Nie, F. Scholer (Eds.), Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019, ACM, 2019, pp. 1409–1410.

[20] S. Zhang, K. Balog, Web table extraction, retrieval, and augmentation: A survey, ACM Trans. Intell. Syst. Technol. 11 (2020).

[21] F. El Outa, et al., Towards a conceptual model for data narratives, in: Conceptual Modeling, Springer International Publishing, Cham, 2020, pp. 261–270.

[22] P. Vassiliadis, P. Marcel, S. Rizzi, Beyond roll-up's and drill-down's: An intentional analytics model to reinvent OLAP (long-version), CoRR abs/1812.07854 (2018). arXiv:1812.07854.

[23] P. Vassiliadis, P. Marcel, S. Rizzi, Beyond roll-up's and drill-down's: An intentional analytics model to reinvent OLAP, Inf. Syst. 85 (2019) 68–91. URL: https://doi.org/10.1016/j.is.2019.03.011. doi:10.1016/j.is.2019.03.011.

[24] E. Wu, Systems for human data interaction (keynote), in: D. Mottin, et al. (Eds.), Proc of the 2nd Workshop on Search, Exploration, and Analysis in Heterogeneous Datastores (SEA-Data 2021@VLDB'21), 2021.

[25] Z. Shang, et al., Davos: A system for interactive data-driven decision making, Proc. VLDB Endow. 14 (2021) 2893–2905.