Managing Data in a Big Financial Institution: Conclusions from a R&D Project

industrial paper

Mariusz Sienkiewicz
Poznan University of Technology
Poznań, Poland
mariusz.sienkiewicz@doctorate.put.poznan.pl

Robert Wrembel
Poznan University of Technology
Poznań, Poland
robert.wrembel@cs.put.poznan.pl

ABSTRACT

Financial institutions (FIs) use the state-of-the art data management and data engineering solutions to support their day-to-day businesses. They follow strict data governance policies as well as country and international regulations. In spite of these facts, the quality of their data is not perfect. Experts in the field estimate that from 1% to about 5% of data owned by financial institutions are dirty. Typically, FIs include in their IT architectures from dozens to a few hundreds of data sources that are being integrated in multiple data warehouses. Such complex architectures generate substantial monetary costs and they are difficult to manage. FIs compete in the financial services market. One way to gain a competitive advantage is to apply the latest technologies for the purpose of data management and shortening a software development cycle. A promising direction is to migrate on-premise infrastructures into private, public, or private-public cloud architectures. In this paper we present our experience from preparing and running a project for a big financial institution in Poland. The project is run in two stages: (1) building a central repository of customers data and (2) developing a data lake architecture in a private-public cloud.

1 INTRODUCTION

Financial institutions (FIs) use the state-of-the art data management and data engineering solutions to store and process their data. They apply strict data governance policies, defined by country (e.g., Financial Supervision Commission in Poland) and international (e.g., European Banking Authority) financial regulatory authorities. User applications, before being deployed in operational IT architectures, undergo thorough testing. FIs strive to gain a competitive advantage by constantly providing new products and services, necessitating the use of the latest technologies and shortening the software development process. Despite the care for the quality of the produced software and data governance processes, the quality of data in financial databases is not perfect. Experts in the field estimate that from 1% to about 5% of data owned by FIs are dirty - mainly with missing or erroneous values, duplicated, and outdated.

Duplicates mainly concern customers data, for the following reasons. First, banks buy other banks, with their information systems and data stored there. Second, some banking products (e.g., a checking account and a stockbroker account) require a separate customer instance in a system for each product, even if a real customer is the same. Third, the imperfection of the software and processes used in data governance allow to create separate customer instances in an information system for the same physical customer even if it is not necessary. Fourth, FIs often function in capital groups, in which individual entities have their own customer databases. In order to manage the relationship with a customer at the level of a capital group, deduplication of customer data is necessary. It is worth to mention that core customers data are related to other data, like contact addresses. This way, duplicate customers data cause duplicates of the related data.

Outdated data is the second issue impacting the quality of data in a FI. This problem concerns customers last names (typically caused by name changing after getting married), postal address (caused by moving to another location), phone numbers, and email addresses, to name the most typical cases.

Duplicated and outdated data cause economic loses and deteriorate reputation of a FI. Thus, clean data are necessary for efficiently conducting a business and for proper functioning artificial intelligence technologies (prediction models, natural language processing, chat bots), which are becoming increasingly important in sales and service processes.

The world of data and IT architectures of a FI must ensure: unambiguous identification of customers, ensure security requirements, meet risk needs, and meet the requirements imposed by institutions regulating the market, including counteracting terrorist financing and money laundering. Another important aspect influencing an IT architecture and data governance policies in a FI is virtualization and service automation. The above aspects, on the one hand, require correct, up-to-date data, but on the other hand, the security policies adopted by a FI limit the possibility of entering or modifying data via remote channels.

Typically, FIs include in their IT architectures from dozens to hundreds of data sources (DSs). Efficient processing of data in different database structures, distributed among multiple DSs, requires the application of an integration architecture. An industry standard is a data warehouses (DW) architecture. In this architecture, DSs are integrated in a central repository - data warehouse by the so-called Extract Transform Load (ETL) processes or their alternative ELT variants (a.k.a. data processing workflows, data processing pipelines, or data wrangling [18, 29]).

An ETL process first *extracts* data of interest from multiple DSs. Second, it *transforms*, cleans, and homogenizes the data. Finally it *loads* the data into a DW. The ELT alternative, extracts data from DSs, loads them in their original formats into an intermediate storage (called an operational data store, data stage, or staging area) and then transforms the data and loads them into a DW. Traditional IT architectures are build based on multiple stand-alone servers and/or data warehouse appliances (e.g., IBM Pure Data for Analytics - Netezza, Oracle Exadata, SAP Hana, Teradata). ETL processes are run by dedicated engines, e.g., [12], typically deployed on a dedicated hardware.

Such complex architectures are difficult to manage from a technological point of view and generate substantial monetary costs. In this context, two following business trends can be observed.

- Leading FIs initiate projects aiming at transforming their on-premise infrastructures into novel architectures - either hybrid or cloud. A *hybrid* architecture includes on-premise databases integrated with databases in a cloud (either private or private-public). Data in both on-premise databases and in a cloud can be accessed and dynamically integrated via a dedicated software layer (with the functionality of a mediated architecture and ETL processes). A pure *cloud* architecture assumes that all on-premise databases have been migrated into a cloud eco-system. In this eco-system, cloud data warehouses are built as well, based on dedicated systems (e.g., Amazon Redshift, Snowflake, Presto, Google BigQuery, Azure Synapse Analytics, Oracle Autonomous Data Warehouse, IBM Db2 Warehouse on Cloud) [33].
- FIs are building cloud repositories of heterogeneous data, typically ingested from sources external to a company, including among others open data published by (local) governments, professional portals, and social media. The data are stored in the repository in their original formats. Such a repository is typically called a data lake (DL) [23, 38]. Further, these data are unified either on-the-fly in the so-called logical data warehouse (LDW) [11, 20, 30] or are homogenized and uploaded into a physical cloud data warehouse (CDW) [16, 31, 37]. Another possible architecture to store and query heterogeneous data is a *polystore*. In this architecture, a few physical data repositories (each of which stores unified data represented in the same data model) are virtually integrated and queried by means of a mediated architecture [5, 10, 17, 41].

Cloud technologies are inevitable also in the financial sector, and the usage of these technologies is already supported by country financial regulatory authorities. For example, in January 2020, the Financial Supervision Commission in Poland approved a document with guidelines and recommendations for deploying cloud services in FIs, thus giving a green light for IT projects based on cloud technologies in the financial sector. In particular, Recommendation D.10.6 *Cooperation with external service providers* (by the Financial Supervision Commission) defines 5 requirements that an external cloud provider must fulfill to be able to offer services for a financial institution.

In this paper, we present our initial experience and challenges in launching a project for a big financial institution in Poland (for this publication, we are not authorized to reveal the name of the institution). Since the project has just started, it is too early to present solutions to the challenges mentioned in this paper. The project is divided into two stages. The first one aims at building a *Central Repository of Customers Data*, integrated from several data sources (cf. Section 2). The second stage aims at building a *Cloud Data Repository* architecture in a *Polish National Cloud*[1] (it is a private-public cloud operated by Microsoft and Google). A few on-premise databases will be next migrated into the repository (cf. Section 3).

[1]https://chmurakrajowa.pl/en/

2 STAGE 1: BUILDING CENTRAL REPOSITORY OF CUSTOMERS DATA

In this stage, data about customers and data related to customers, from multiple data sources, are integrated into the *Central Repository of Customers Data* (CRCD).

2.1 Stage goals

The goals to achieve within this stage are as follows:

G1 - to integrate customers data and related data in the CRCD;
G2 - to homogenize the structures of records describing customers and the structures of related data, coming from multiple data sources;
G3 - to homogenize values of the integrated data;
G4 - to clean data;
G5 - to develop data deduplication piplelines, in two variants: based on statistical models and based on machine learning models;
G6 - to develop machine learning models for customers data aging.

2.2 Architecture

An overall technical architecture being build in this stage is shown in Figure 2.2. It is a standard data warehouse architecture, where goals G1-G4 are implemented by means of ETL processes. Source customers data and related data are stored in relational databases, denoted as DS_1-DS_6; the *CRCD* is also a relational database (Oracle DBMS). In order to support data cleaning and standardization, ETL processes use open data sources (denoted as *openDS*) and paid data sources (denoted as *paidDS*). Both types of DSs are made available by the public administration. These sources include reference data, among others on: citizen unique IDs, administrative division of a country, zip codes, and companies.

An important functionality of the ETL layer is a *data deduplication pipeline* (DDP). The DDP realizes goal G5. Having profiled the customers data in the available source systems, we conclude that the DDP cannot be fully automatized and there are cases where an expert knowledge is required to support the process, however, the DDP is to minimize the number of rows requiring manual work.

Being processed by the ETL processes, customers and related data are uploaded into the *CRCD*. The usage of the *CRCD* is twofold. First, its content is to be analyzed by data mining algorithms in order to discover models for data aging (goal G6). The overall idea behind this component is to be able to discover classes of data and their properties that share similar (or identical) aging characteristics. Second, the *CRCD* will become a source of truth for the whole data infrastructure of the FI, thus it will be accessed by other internal systems of the FI.

2.3 Challenges

While designing the aforementioned architecture, even though it is a standard one, we encountered a few issues. The most challenging ones include: (1) designing a data deduplication pipeline and (2) developing ML models for predicting data aging.

2.3.1 Designing data deduplication pipeline. An important and challenging task in the ETL is a data deduplication pipeline. The state of the art pipeline [8, 26] is shown in Figure 2.3.1. It is composed of four main tasks, namely *blocking*, *block processing*, *entity matching*, and *entity clustering*. Blocking aims at dividing

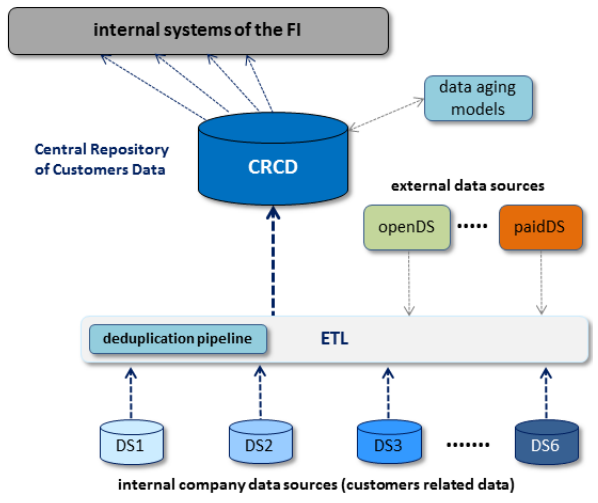
internal systems of the FI
data aging models
Central Repository of Customers Data
CRCD
external data sources
openDS
paidDS
deduplication pipeline
ETL
DS1
DS2
DS3
DS6
internal company data sources (customers related data)

Figure 1: The overall architecture of the *Central Repository of Customers Data*

into groups records that are likely to represent duplicates, with the final goal to reduce the number of records that need to be compared. In this DDP, records are compared only within blocks they belong to, which can be done in parallel. Block processing aims at reorganizing records in blocks, in order to further reduce the number of needed comparisons. Entity matching consists in computing similarity measures between records. Finally, in the entity clustering step, identical records (i.e., those whose similarity measures, e.g., [24, 42] are higher than a given threshold) are grouped together and merged.

Even though, research on deduplication has been done for decades, and multiple algorithms have been developed for each step in the DDP, the whole pipeline has to be constructed manually. The main problem remains in selecting the most adequate algorithm to run each of the four tasks, so that the pipeline maximizes precision and recall. To fully solve this problem, one would need to apply a greedy approach of testing the results of all possible combinations of these algorithms. However, it is not feasible as it is an optimization problem of an exponential complexity. For example, in *block building* there are at least 14 popular algorithms, in *block processing* there are at least 18 algorithms, in *entity matching* there are at least 20 algorithms, and in *entity clustering* there are at least 7 algorithms [9, 25–27], resulting in at least 35280 combinations.

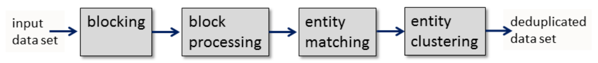

Figure 2: The state of the art data deduplication pipeline

The algorithms in the DDP assume that the processed data have been cleaned and standardized earlier. In the case of customer data, such cleaning and standardization is not possible for first and last names, as there are multiple variants of standard names. For example, names like 'Anna' and 'Ana' may be treated as being the same, assuming that a typo was done, or as being two different names; a non-Latin name may be translated into the Latin alphabet in multiple ways.

Once the entity clustering step has identified records representing the same customer, a reference customer record needs to be build based on these records. If, however, the records differ with values of addresses, a problem arises in figuring out which address it the current one. Typically, changes in data are timestamped and in such a case, the most recently timestamped address and other contact data may be used as the current ones. Unfortunately, in the reality, the most recent timestamp may be old. In such a case, it is probable that some data may have changed since then. This observation led us to the conclusion that data aging models could help solving also this problem (cf. Section 2.3.2.

To conclude, having done the analysis of the state of the art in designing the DDP, to the best of our knowledge, we conclude that an automatic approach to constructing an end-to-end data deduplication pipeline has not been proposed yet for traditional record-like data. The problem is getting more difficult for the deduplication of big data, mainly, because big data are represented in a plenthora of different formats. Before applying the DDP, all these data must be unified into the same format, which itself is a challenge. We encountered this problem in the second stage of our project (cf. Section 3).

2.3.2 Developing ML models for data aging. An intrinsic feature of some types of data, is their aging. Four main types of such data are of special interests by FIs, namely: (1) customers last names, (2) customers identification documents, (3) postal addresses, and (4) contact addresses (phone numbers, emails). Such data become outdated mainly as the result of: last name changing after getting married or divorced, expiry dates reached by identification documents, customers moving to other locations, as well as changing phone numbers and email addresses.

Outdated data decrease reliability of data and cause financial loses. For these reasons, FIs so fare have been using analog methods to keep customers data up to date (e.g., checking data upon a customer arrival to a FI branch, checking data of delivered and not deliver mailings, calling a customer to verify her/his data). Moreover, FIs strive to ensure the highest possible level of customer self-service by means of remote services via the Internet, which results in limited contact between employees of a branch network and customers. Therefore, the analog methods of verifying the correctness and timeliness of customer data either do not work or are inefficient in therms of costs, speed, and the amount of data that can be updated.

For this reason, in this project we aim at developing data aging models based on ML techniques. We assume that dedicated models will be built at least for a few age groups of customers. To the best of our knowledge, such models have not been proposed yet in the context of our project. The only approach addressing a related problem is [44], but in the context of temporal data. Other approaches address the problem of moving cold data from a hot to a cold storage, e.g., [32] or to an external storage, e.g., [7].

At this stage of the project development, we plan to start experimenting with classification for building data aging models. It is very likely that data imbalance will require the application also of other ML techniques, like neural networks.

3 STAGE 2: BUILDING CLOUD DATA REPOSITORY

In this stage, the so-called *Cloud Data Repository* (CDR) is build (notice that such a repository has features of a data lake). The *CDR* will include: (1) the *CRCD* developed in Stage 1 (cf. 2), (2) the data sources storing data related to customers records, and (3) external data sources augmenting views on customers, to achieve functionality of *customer 360*.

3.1 Stage goals

The goals to achieve in this stage are as follows:

G7 - to build the *CRD* for storing data from selected internal company DSs;
G8 - to augment customers view with data from external DSs (professional portals and social portals);
G9 - to design data retention models in a cloud eco-system;
G10 - to develop a data governance method in a cloud eco-system; the method must be compliant with national and international regulations in the financial sector;
G11 - to develop an end-to-end method for designing and deploying a data repository in a cloud eco-system.

3.2 Architecture

The aforementioned goals will be achieved in the architecture that we have designed and will be implementing. The architecture is shown in Figure 3.2. It is a *hybrid* cloud eco-system, where some DSs are stored in an on-premise architecture ($DS_1, \ldots, DS_i$ and *CRCD*) and some DSs are stored in a private-public cloud eco-system.

The core component of the hybrid cloud eco-system is the *Cloud Data Repository*. It includes:

- internal company DSs, denoted as $IDS_j, \ldots, IDS_m$, and *CRCD*; these DSs will be migrated from the on-premise architecture into the *CDR*; the sources are relational databases and they store customers and related data; notice that the *CRCD* developed in the first stage is also integrated into the *CDR*;
- external DSs, denoted as EDS_n, EDS_o, and EDS_p; EDS_n stores data ingested from open data provided by the public administration; EDS_o stores data ingested from commercial repositories (paid) provided by the public administration; EDS_p stores data ingested from professional and social media sources in the Internet and internal customer behavioral data; EDS_n, EDS_o, and EDS_p store data related to customers (both individuals and companies); these DSs are non-relational, typically HTML; XML; JSON, and RDF;
- a data warehouse, denoted as *DW*, integrating customers and related data from: (1) $IDS_j, \ldots, IDS_m$, (2) *CRCD*, and (3) EDS_n, EDS_o, EDS_p, cleaned, deduplicated, and unified into a common data format; the content of the *DW* will become the source of truth about customers for the on-premise databases in the FI.

Data from DSs in the *Cloud Data Repository* will be integrated into the *DW* by means of ETL/ELT processes implemented in the cloud eco-system. The *DW* will be periodically refreshed with customers data from the *CRCD* by means of ETL processes implemented in Informatica. Since the *CRCD* is replicated into the *DW*, the *CRCD* will be maintained in the on-premise architecture only until the *CDR* is fully operational. After achieving a fully operational architecture, the *CRCD* will be replaced by the *DW* in the *CDR*.

3.3 Challenges

While designing the aforementioned architecture we encountered the following challenges: (1) designing efficient ETL/ELT processes, (2) handling the evolution of data sources, (3) designing logical and physical data lake schemas for storing unstructured data, and (4) provisioning optimal cloud resources.

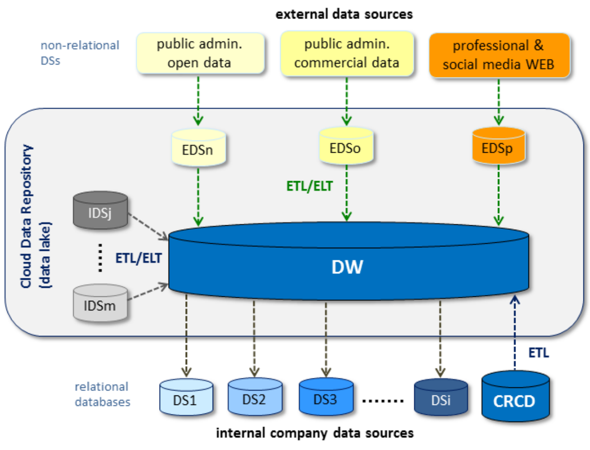

Figure 3: The overall architecture of the *Cloud Data Repository*

3.3.1 Designing efficient ETL processes. Typically, in a standard DW architecture, all ETL processes must finish within a given time window, usually within a few hours, to make a DW available for analtytics. Therefore, assuring efficient executions of ETL processes (typically measured by throughput and an overall execution time) is challenging [1]. In a DL architecture, this task is much more challenging due to two main reasons. First, ETL processes in a DL architecture ingest and process larger data volumes than in a standard DW architecture. Second, the variety and complexity of data models and formats that ETL processes must process is much larger in a DL architecture.

In practice, the performance of an ETL process may be improved by: (1) scaling up or out hardware on which the process is run, (2) running the process in parallel, and (3) reordering its tasks [19, 34]. Existing commercial ETL/ELT tools provide means of parallelizing ETL tasks, but it is the responsibility of an ETL developer to select tasks for parallelization and to apply an appropriate parallelization skeleton. Reordering ETL tasks may reduce the execution time of an ETL process, but finding a reordering that would yield the shortest execution time is of exponential complexity [35]. Again, either commercial or free ETL engines do not offer any means for automatic optimization of ETL processes by tasks reordering, with the exception of IBM InfoSphere and Informatica PowerCenter, which allow to move some tasks into a data source to be executed there [14, 21].

To sum up, the performance of an ETL process largely depends on a manual orchestration and parallelization of tasks within an ETL process by an ETL developer.

At this stage of the project, we plan to apply a known heuristic to place the most restrictive tasks, i.e., those which filter data, at the beginning of an ETL process, in order to reduce a data volume ingested by an ETL process as soon as possible. Another promising direction would be to allocated adequate cloud resources to run a given ETL process under a monetary or time constraint budget, in the spirit of [2].

3.3.2 Handling evolution of DSs. An intrinsic feature of internal company DSs and external DSs is the evolution of their structures (schemas) in time [39, 40]. Internet data sources evolve much frequently than internal company DSs. Such evolution has an impact on a data integration layer, i.e., ETL processes ingesting data from evolving DSs stop working with errors.

As part of the preparation to the project described in this paper, we had run a pilot micro-project aiming at creating a micro-DL with data ingested from Internet data sources. The micro-DL stored: (1) data about companies being customers of the FI and (2) descriptions of financial products offered by banks in Poland. To this end, we integrated: (1) a few professional portals providing data about companies (including LinkedIn, Glassdoor, GoldenLine) and (2) several portals of banks offering services in Poland, to compare their product offers.

We applied an ELT architecture in two alternative cloud eco-systems, namely GCP and AWS. In both cases we experienced problems with fast (sometimes day-to-day) changing structures of data provided by the aforementioned Internet data sources. As a consequence, previously designed and deployed ELT processes generated errors and needed to be repaired, i.e., adjusted to new structures of the changed DSs. This had to be done manually as neither of the commercial and free ETL/ELT tools supports an automatic repair of such processes. It is one of the still open problems in DW/DL research [6, 43].

The second problem caused by evolving DSs is related to detecting data changes and structural changes. A DW or DL are typically refreshed incrementally. To this end, an ETL/ELT process has to construct the increment (a.k.a. delta). However, in multiple architectures, the only way to access a DS is to use its data export (a.k.a. snapshot) provided by the DS. Frequently, such a snapshot includes the whole content of the DS. From this content, the ETL/ELT process has to extract the increment. It is done by comparing two consecutive snapshots. When a DS changed its structure, then two consecutive snapshots could not be comparable, thus an increment could not be constructed.

Furthermore, snapshot comparison is challenging for DSs that use complex data structures (e.g., graphs, semi-structured, NoSQL) as the comparison algorithm has to be able to traverse nested structures and handle cases when new nested objects appear.

The micro-project revealed that while dealing with Internet data sources, one cannot use off-the shelf solutions (as they do not exist). In the project, ELT processes were repaired manually. Data increments were constructed by comparing two consecutive snapshots of nested data, at a cost of expensive processing. The snapshot comparison algorithm had to be changed manually when a DS changed its structure.

3.3.3 Designing logical and physical DL schemas. Designing logical and physical schemas for relational DWs is a very well researched topic, supported by mature relational database management systems as well as by design and development tools. DW modeling is a fundamental task being done in an early phase of a DW development. Modeling data lakes still needs to be researched. Dealing with unstructured data tempts a designer to apply NoSQL storage, to model a flexible schema. Unfortunately, NoSQL storage servers are not mature yet and they do not offer a rich SQL syntax, advanced indexing, and advanced cost-based query optimization. On the contrary, relational database management systems (especially the commercial ones) offer such a features at the cost of rigid schemas.

Despite of the fact that research is being conducted on data lake modeling (i.e., for unstructured data), e.g., [13, 15], there is no commonly accepted modeling method. Research and development in physical storage, physical data structures, e.g., [28, 45], and query optimization techniques for NoSQL storage, e.g., [22] is being conducted as well, but there is no single product that would offer all these three important functionalities, which mature commercial RDBMSs do offer.

To conclude, designing logical DL schemas, physical storage, and physical data structures are not guided by any method (like for relational databases) and must be done case by case (ad-hoc).

3.3.4 Provisioning optimal cloud resources. While deploying a cloud architecture, one typically aims at achieving the best possible performance with minimized monetary costs paid for the cloud infrastructure. Different types of processing (e.g., analytical, transactional, ETL) require different amounts of cloud resources (e.g., the number of virtual machines, the number of CPU and cores, main memory, disk storage).

Provisioning optimal resources for a given type of processing to maximize performance is contradictory to minimizing monetary costs and the optimization of these goals is a combinatorial problem of exponential complexity, e.g., [3]. Despite the fact that some research has been and still is conducted in this area, e.g., [4, 29, 36], it is considered for the time being as an open problem.

4 SUMMARY

In this paper we outlined a project being done for a big financial institution in Poland. Its goal is to build a system providing clean data about customers and their related data, augmented from external data sources. To this end, a hybrid architecture is built for the FI. The architecture is composed of the on-premise databases and the private-public cloud eco-system. The cloud eco-system will include customers data and data related to customers, migrated from the old on-premise architecture and data ingested from Internet data sources.

The focus of this paper is on presenting the hybrid architecture that we designed and on challenges that we encountered while realizing the project (since the project has just started, we are not able to provide solutions to the challenges yet).

Based on the gained early experience and based on the state of the art analysis in research and technology, we can draw the following conclusions:

- a commonly shared knowledge on building data lakes in a cloud for a FI is not available (as the most probably it is considered as a company's asset);
- comprehensive methods for guiding the process of efficient data migration from an on-premise to a cloud architecture are not available either;
- a comprehensive method for building logical and physical DL models are not available either;
- a reference DL architecture in a cloud for a FI is not available (as the most probably it is considered again as a company's asset);
- finally, an end-to-end method for designing and deploying a data lake for a FI (from conceptual, logical, and physical modeling, through data migration and ingestion, data governance, to performance optimization) still needs to be developed.

Moreover, running a project for a big FI requires usage of commercial tools at every development stage and for every task of a DW and DL development step. For a big FI, the only applicable solution is to use powerful commercial software. For example, commercial ETL engines offer rich functionalities, including parsing addresses and names, algorithms ready to use, cleaning techniques and some deduplication techniques, parallel processing either in on a single server or in a cloud, accessing non-relational data. Also technical support from a software house is of great importance. In the described project we use Informatica for data profiling and ETL; Oracle databases - for storing internal company data, including the *CRCD*; Microsoft Azure and Google Cloud Platform - as private-public cloud eco-systems in the *Polish National Cloud*.

REFERENCES

[1] Syed Muhammad Fawad Ali and Robert Wrembel. 2017. From conceptual design to performance optimization of ETL workflows: current state of research and open problems. *The VLDB Journal* 26, 6 (2017), 777–801.

[2] Syed Muhammad Fawad Ali and Robert Wrembel. 2019. Towards a Cost Model to Optimize User-Defined Functions in an ETL Workflow Based on User-Defined Performance Metrics. In *European Conf. Advances in Databases and Information Systems (LNCS, Vol. 11695)*. Springer, 441–456.

[3] Syed Muhammad Fawad Ali and Robert Wrembel. 2020. Framework to Optimize Data Processing Pipelines Using Performance Metrics. In *Int. Conf. on Big Data Analytics and Knowledge Discovery (DaWaK) (LNCS, Vol. 12393)*. Springer, 131–140.

[4] Abdullah Khalid A. Almasaud, Agresh Bharadwaj, Sandra Sampaio, and Rizos Sakellariou. 2020. Challenges in Resource Provisioning for the Execution of Data Wrangling Workflows on the Cloud: A Case Study. In *Int. Conf. on Database and Expert Systems Applications (DEXA)*. LNCS 12392, 66–75.

[5] Rana Alotaibi, Damian Bursztyn, Alin Deutsch, Ioana Manolescu, and Stamatis Zampetakis. 2019. Towards Scalable Hybrid Stores: Constraint-Based Rewriting to the Rescue. In *SIGMOD*. 1660–1677.

[6] Judith Awiti and Robert Wrembel. 2020. Rule Discovery for (Semi-)automatic Repairs of ETL Processes. In *Int. Baltic Conf. on Databases and Information Systems (CCIS, Vol. 1243)*. Springer, 250–264.

[7] Arun Balasubramanyan. 2014. Data warehouse augmentation, Part 3: Use big data technology for an active archive. https://www.ibm.com/developerworks/library/ba-augment-data-warehouse3/index.html. IBM DeveloperWorks.

[8] Vassilis Christophides, Vasilis Efthymiou, Themis Palpanas, George Papadakis, and Kostas Stefanidis. 2019. End-to-End Entity Resolution for Big Data: A Survey. *CoRR* abs/1905.06397 (2019).

[9] Adrian Colyer. 2020. The morning paper on An overview of end-to-end entity resolution for big data. https://blog.acolyer.org/2020/12/14/entity-resolution/.

[10] Jennie Duggan, Aaron J. Elmore, Michael Stonebraker, Magdalena Balazinska, Bill Howe, Jeremy Kepner, Sam Madden, David Maier, Tim Mattson, and Stanley B. Zdonik. 2015. The BigDAWG Polystore System. *SIGMOD Rec.* 44, 2 (2015), 11–16.

[11] Ted Friedman and Nick Heudecker. 2020. Data Hubs, Data Lakes and Data Warehouses: How They Are Different and Why They Are Better Together. Gartner.

[12] Gartner. 2019. Magic Quadrant for Data Integration Tools.

[13] Corinna Giebler, Christoph Gröger, Eva Hoos, Holger Schwarz, and Bernhard Mitschang. 2019. Modeling Data Lakes with Data Vault: Practical Experiences, Assessment, and Lessons Learned. In *Int. Conf. on Conceptual Modeling ER (LNCS, Vol. 11788)*. Springer, 63–77.

[14] Informatica. 2007. How to Achieve Flexible, Cost-effective Scalability and Performance through Pushdown Processing. Whitepaper.

[15] Matthias Jarke and Christoph Quix. 2017. On Warehouses, Lakes, and Spaces: The Changing Role of Conceptual Modeling for Data Integration. In *Conceptual Modeling Perspectives*. Springer, 231–245.

[16] Sean Michael Kerner. 2019. Top 8 Cloud Data Warehouses. https://www.datamation.com/cloud-computing/top-cloud-data-warehouses.html. Datamation.

[17] Boyan Kolev, Carlyna Bondiombouy, Patrick Valduriez, Ricardo Jiménez-Peris, Raquel Pau, and José Pereira. 2016. The CloudMdsQL Multistore System. In *SIGMOD*. 2113–2116.

[18] Nikolaos Konstantinou and Norman W. Paton. 2020. Feedback driven improvement of data preparation pipelines. *Inf. Syst.* 92 (2020), 101480.

[19] Nitin Kumar and P. Sreenivasa Kumar. 2010. An Efficient Heuristic for Logical Optimization of ETL Workflows. In *VLDB Workshop on Enabling Real-Time Business Intelligence*. 68–83.

[20] Alice LaPlante. 2020. Building a Unified Data Infrastructure. O'Reilly whitepaper.

[21] Rao Lella. 2014. Optimizing BDFS jobs using InfoSphere DataStage Balanced Optimization. IBM DeveloperWorks.

[22] Divya Mahajan, Cody Blakeney, and Ziliang Zong. 2019. Improving the energy efficiency of relational and NoSQL databases via query optimizations. *Sustain. Comput. Informatics Syst.* 22 (2019), 120–133.

[23] Fatemeh Nargesian, Erkang Zhu, Renée J. Miller, Ken Q. Pu, and Patricia C. Arocena. 2019. Data Lake Management: Challenges and Opportunities. *Proc. VLDB Endow.* 12, 12 (2019), 1986–1989.

[24] Felix Naumann. 2013. Similarity measures. Hasso Plattner Institut.

[25] George Papadakis and Themis Palpanas. 2018. Web-scale, Schema-Agnostic, End-to-End Entity Resolution. In *Tutorial at World Wide Web*.

[26] George Papadakis, Dimitrios Skoutas, Emmanouil Thanos, and Themis Palpanas. 2020. Blocking and Filtering Techniques for Entity Resolution: A Survey. *ACM Comput. Surv.* 53, 2 (2020), 31:1–31:42.

[27] George Papadakis, Leonidas Tsekouras, Emmanouil Thanos, George Giannakopoulos, Themis Palpanas, and Manolis Koubarakis. 2019. Domain- and Structure-Agnostic End-to-End Entity Resolution with JedAI. *SIGMOD Record* 48, 4 (2019), 30–36.

[28] Philipp D. Rohde and Maria-Esther Vidal. 2020. Optimizing Federated Queries Based on the Physical Design of a Data Lake. In *Proc. of EDBT/ICDT Workshops (CEUR Workshop Proceedings, Vol. 2578)*.

[29] Oscar Romero and Robert Wrembel. 2020. Data Engineering for Data Science: Two Sides of the Same Coin. In *Proc. of Int. Conf. on Big Data Analytics and Knowledge Discovery (DaWaK) (Lecture Notes in Computer Science, Vol. 12393)*. Springer, 157–166.

[30] Philip Russom. 2019. Modernizing the Logical Data Warehouse. https://tdwi.org/articles/2019/10/14/dwt-all-modernizing-the-logical-data-warehouse.aspx. TDWI.

[31] John Ryan and Uli Bethke. 2019. A Comparison of Cloud Data Warehouse Platforms. https://www.datamation.com/cloud-computing/top-cloud-data-warehouses.html. Sonora Intelligence.

[32] SAP. [n.d.]. Data aging. SAP Help portal.

[33] ScienceSoft. [n.d.]. Data Warehouse in the Cloud: Features, Important Integrations, Success Factors, Benefits and More. https://www.scnsoft.com/analytics/data-warehouse/cloud.

[34] Alkis Simitsis, Panos Vassiliadis, and Timos K. Sellis. [n.d.]. Optimizing ETL Processes in Data Warehouses. In *Int. Conf. on Data Engineering ICDE*. 564–575.

[35] Alkis Simitsis, Panos Vassiliadis, and Timos K. Sellis. 2005. State-Space Optimization of ETL Workflows. *IEEE Transactions on Knowledge and Data Engineering* 17, 10 (2005), 1404–1419.

[36] Sukhpal Singh and Inderveer Chana. 2016. Cloud resource provisioning: survey, status and future research directions. *Knowl. Inf. Syst.* 49, 3 (2016), 1005–1069.

[37] Mohamed A. Soliman, Lyublena Antova, Marc Sugiyama, Michael Duller, Amirhossein Aleyasen, Gourab Mitra, Ehab Abdelhamid, Mark Morcos, Michele Gage, Dmitri Korablev, and Florian M. Waas. 2020. A Framework for Emulating Database Operations in Cloud Data Warehouses. In *Proc. of SIGMOD*. ACM, 1447–1461.

[38] Ignacio Terrizzano, Peter Schwarz, Mary Roth, and John E. Colino. 2015. Data Wrangling: The Challenging Journey from the Wild to the Lake. In *Conf. on Innovative Data Systems Research (CIDR)*.

[39] Panos Vassiliadis and Apostolos V. Zarras. 2017. Schema Evolution Survival Guide for Tables: Avoid Rigid Childhood and You're En Route to a Quiet Life. *J. Data Semant.* 6, 4 (2017), 221–241.

[40] Panos Vassiliadis, Apostolos V. Zarras, and Ioannis Skoulis. 2017. Gravitating to rigidity: Patterns of schema evolution - and its absence - in the lives of tables. *Inf. Systems* 63 (2017), 24–46.

[41] Marco Vogt, Alexander Stiemer, and Heiko Schuldt. 2018. Polypheny-DB: Towards a Distributed and Self-Adaptive Polystore. In *BigData*. 3364–3373.

[42] Jiannan Wang, Guoliang Li, Jeffrey Xu Yu, and Jianhua Feng. 2011. Entity Matching: How Similar Is Similar. *VLDB Endow.* 4, 10 (2011), 622–633.

[43] Artur Wojciechowski and Robert Wrembel. 2020. On Case-Based Reasoning for ETL Process Repairs: Making Cases Fine-Grained. In *Int. Baltic Conf. on Databases and Information Systems (CCIS, Vol. 1243)*. Springer, 235–249.

[44] Anita Zakrzewska and David A. Bader. 2016. Aging data in dynamic graphs: A comparative study. In *Int. Conf. on Advances in Social Networks Analysis and Mining, ASONAM*. IEEE Computer Society, 1055–1062.

[45] Dongfang Zhang, Yong Wang, Zhenling Liu, and Shijie Dai. 2019. Improving NoSQL Storage Schema Based on Z-Curve for Spatial Vector Data. *IEEE Access* 7 (2019), 78817–78829.