A System for Collecting, Managing, Analyzing and Sharing Diverse, Multi-faceted Cultural Heritage and Tourism Data

Kimon Deligiannis*, Paraskevi Raftopoulou†, Christos Tryfonopoulos‡ and Costas Vassilakis§
Department of Informatics & Telecommunications,
University of the Peloponnese,
Tripolis, GR22100, Greece
Email: *deligiannis@uop.gr, †praftop@uop.gr, ‡trifon@uop.gr, §costas@uop.gr

2021 16th International Workshop on Semantic and Social Media Adaptation & Personalization (SMAP) | 978-1-6654-4241-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/SMAP53521.2021.9610777

Abstract—**Today, social media platforms and other online sources, like forums and review sites, offer an abundance of cultural and touristic information that is voluntarily offered by travelers; this information, although helpful for other travelers, is typically fragmented and thus cannot be easily leveraged to exploitable knowledge by scientists and other tourism stakeholders. In this work, we present a novel, integrated system for collecting, managing, analyzing and sharing diverse, multi-faceted cultural heritage/tourism-related data that aims to assist scientists in the cultural heritage domain and tourism stakeholders to gather and synthesize scattered information to exploitable knowledge. The proposed system is tailored to the tourism domain needs, and allows users with minimum effort and zero IT expertise to (i) gather data from both structured and unstructured/semi-structured online sources, (ii) leverage the data to knowledge via appropriate analysis and visualization tools, and (iii) share the collected data and gathered knowledge with other stakeholders via appropriate publish-subscribe mechanisms. The proposed system is entirely open-source, designed upon big data tools and principles for the data store, the analytics production, and the knowledge sharing, and targets both performance and usability.**

I. Introduction

In the Social Media (SM) and the Internet of Things (IoT) era, the vast amounts of data that are produced every day and the increasing number of people who register in SM platforms turn out to be two interconnected notions [1], [2], [3], [4]. SM users are characterized as *prosumers* -i.e. producers *and* consumers, since they not only benefit from utilizing the SM services, but also produce and publish content themselves. This content eventually becomes essential for economy sector applications and for other institutional operations in many scientific domains [5]. In the same manner, SM users produce immense amounts of information in the Cultural Heritage (CH) and tourism domains by generating varying types of data, including photos, videos, reviews or own stories, trajectories, geospatial data, URLs linking items etc. A considerable amount of this data are posted on social networks through the users' IoT devices, mainly smartphones; the information is either posted on their own profile page or on the page of a cultural venue [6]. This aggregation of heterogeneous SM data flows composes an information waterfall that can be defined by the "3Vs" [7] that represent the *Volume*, the *Velocity* and the *Variety* of Big Data. Such an information plethora can be used in various scientific and domain-specific applications in the CH and tourism fields: scientists in the Cultural Informatics (CI) and the tourism domains can capture, manage and analyze this data, with the view to synthesize and exploit knowledge of high importance, which can then be used to improve the way visitors experience cultural venues (e.g. archaeological sites, museums, galleries and other cultural foundations) [8], [9], [10]. However, as the magnitude and the diversity of information increases, many stakeholders associated with the CI and tourism domains realize that the traditional information management approaches are inadequate [11]. This does not only stem from the 3Vs that characterize cultural data, but also from the diversity in the needs of data consumers, with each consumer category requiring a different viewpoint (or *facet*) to the data collection.

This paper presents a novel, integrated system for collecting, managing, analyzing and sharing diverse, multi-faceted cultural heritage/tourism-related data. The proposed system can gather data from both structured and unstructured/semi-structured sources, and stores all data under a homogenized scheme in a flexible, document-oriented store. Users may access the data either on-request, through data analytics and visualization services, or according to a publish-subscribe scheme. The proposed system also includes functionality for administrators to manage users and access rights to the content. The system is designed to require low or no IT expertise for deploying, populating and managing data collections, facilitating and accelerating the relevant operations.

The rest of the paper is structured as follows: section II overviews related work on information systems providing flexible document storage. Section III presents the proposed system, while in section IV conclusions are drawn and future work is outlined.

II. Related Work

In recent years, with the emergence of big data, considerable research efforts on information systems have been conducted, aiming to improve and facilitate the harvesting, storage, querying and analysis of vast quantities of structured, semi-structured and unstructured information. However, the idiosyncrasies and heterogeneity of data, technology evolution, scalability demand as well as evolving and fluid user requirements necessitate pioneering and efficient scientific approaches. In this section, we overview the related state-of-the-art literature that offers solutions to the aforementioned issues. This literature is classified into two parts; the first part is related to the NoSQL (Not Only SQL) database systems philosophy, which is the new wave of high-performance database systems, tailored to meet the expanding requirements of modern big data applications. The second part focuses on the most notable approaches in information systems that employ the document-oriented NoSQL paradigm to realize the data storage layer.

A. NoSQL database approaches

As the variety and the mass of data produced by web applications, SM and the IoT grows every minute, the need of novel database management techniques, that are able to efficiently adapt and manage this data supporting the needs of information systems rises too. The solutions presented in [12], [13], [14], tackle this issue by putting aside the conventional RDMS (Relational Database Management Systems) SQL-based approaches and introducing the most widespread NoSQL implementations. The NoSQL databases surveyed in these works are classified in four main categories, namely (i) Key-Value, (ii) Wide-Column, (iii) Document-oriented and (iv) Graph-oriented databases), and each database class is evaluated in terms of scalability, performance, consistency, security, analytical capabilities and fault-tolerance, to conclude that each NoSQL database kind can handle different operations better, whereas the database selection has to be determined depending on the use case scenario and the organization/application needs. The authors in [15] state that the column-based NoSQL DBMS are not able to support online analysis operators (OLAP) and suggest a cube operator, coined MC-CUBE (MapReduce Columnar CUBE), which enables the construction of columnar NoSQL cubes once collecting the data repositories. Two more column-oriented NoSQL paradigms are presented in [16], [17]. The first work proposes a method to put on conversion regulations in order to migrate the SQL relational database content to a big data column-based NoSQL database, while the second one is associated with the comprehension, the arrangement, the pros and cons of Clickhouse database and the way that this column-oriented NoSQL database is able to replace Oracle relational database when the workload augments. In [18], the authors argue that database schemas need to be transformed to meet ever-changing application needs. To tackle this issue, they introduce a framework to detect alterations in a NoSQL database schema and its data, supporting efficiently the conceptual model evolution. The contribution in [19], examines the way to implement a big data mart on a key-value based NoSQL database, by applying a transformation procedure from a multifaceted conceptual schema to a logical pattern, employing three models that furnish key-value stores with an SQL-like table structure overlay. In [20], the authors present a data lake, designed to store multi-sourced structured and unstructured data streams characterized by the 3Vs [7], that is implemented utilizing the Hadoop Distributed File System (HDFS) on the Hadoop Data Platform (HDP) and they apply it on a car trading company use case scenario.

Regarding the management of geographical content derived from the Web, which is a major application class involving SM and IoT systems, the approaches in [21], [22] focus on NoSQL DBMSs capable of storing and indexing spatio-temporal and geospatial data respectively. The first solution, named TrajMesa, is a route storage system which is founded on GeoMesa and embraces an innovative warehousing mechanism that minimizes the storage volume and is able to support queries adeptly, while GeoYCSB constitutes a benchmarking framework designed to measure the performance and scalability between NoSQL databases, capable to manage geospatial volumes of work.

The necessity of decision-making and knowledge extraction derived from data generated from social networking platforms, that leads to Extract-Transform-Load (ETL) big data procedures development is discussed in [23]. The presented solution, coined BigDimETL (Big Dimensional ETL), has been assembled by exploiting MapReduce and Hbase technologies and is closely aligned with ETL methods adaptation. A prominent work concerning document-oriented NoSQL databases is presented in [24]. In this work, the authors, propose a database schema recommendation model at the primary system evolution phase, according to the use case scenario requirements and CRUD functionalities, that is capable to assist developers in the context of a time consuming process, such as designing a NoSQL database. Finally, [25] asserts that the implementation of composite queries on multiform data stores is very challenging; to this end, the authors introduce an intermediate VDS (Virtual Data Store) module, able to efficiently perform complex queries on several heterogeneous data stores in Cloud environments.

B. NoSQL information systems

Big data analytics have emerged as a significant research domain that necessitates a number of requirements from the data storage layer, including performance, scalability, flexibility and manageability. The need for scalable applications, capable to support heavy workloads and retrieve data effectively has led many systems to adopt document-based NoSQL databases [26]. This section presents the state-of-the-art approaches that make use of the NoSQL paradigm to realize the data storage layer.

In the approaches cited in [27], [28], the authors focus on knowledge representation through ontologies, a modeling tool that is frequently used in the data integration research domain. To this end, they follow a three-stages procedure by stockpiling and homogenizing the data to a NoSQL database, spawning local ontologies and finally, arranging the local ontologies to produce a more generic one. In [29], a novel method combining both a routing meta-model and a number of innovative algebraic structures to provide a real-time analysis of traffic jams in urban environments is demonstrated. The approach takes advantage of space-time path data collected using cloud computing and IoT technologies, while data storage and management is achieved using a NoSQL database and in Hadoop ecosystem respectively. The authors in [30] introduce a Mongoose mediation module which is able to model and manage real-time temporal data, obtained from ANT+ sensors, as hierarchical MongoDB documents in a Node.js environment. [31] present a framework aiming to support and facilitate the analysis of semi-structured application data stored in a NoSQL database, while delivering high performance; the framework is applied in a cluster monitoring and prediction use case. In a similar spirit, MyStore [32] is an efficient, user-friendly and always-available dispersed storage framework for handling large amounts of unstructured data using NoSQL. CryptMDB [33], constitutes a cumulative homomorphic asymmetric cryptosystem capable to encipher users' data maintained in a NoSQL database. The approach in [34], puts forward a user-friendly tool performing model transmutation from relational databases to NoSQL for automatic data migrations.

The work in [3] demonstrates a data lake employing NoSQL with a view to store, manage and bridge vast and diverse data sets embracing heterogeneous information from multiple SM sources. This approach is the most theoretically and operationally similar work to the one presented in this paper; however, the SM data extraction in this work is achieved by retrieving information from the API directly or by using third party applications in order to interact with the related SM's API, while data acquisition in our work is realized by initiating properly adapted SM crawlers that are specifically developed for this purpose; furthermore, the work in [3] was designed for the needs of a specific project and applied to a particular research study, while our work is an online, free, zero-administration data lake that offers both fundamental and advanced user and data/knowledge management functionality in the CH and the tourism domains, able to be customized for the requirements of any CH or tourism-related project, and addresses all users, without requiring any IT background/skills.

Large volumes of data can be also found in the e-learning domain, especially in the pandemic era where students participate in their courses remotely, producing thus a bigger digital footprint. The e-learning infrastructure cited in [35], adopts a hybrid database architecture encapsulating both RDBMS for managing structured data and a NoSQL database for manipulating unstructured data, while SCeLE [36] has been restructured to use MongoDB (a NoSQL implementation) instead of MySQL, to prove much faster.

The necessity of big data technologies has emerged in the research field of EHR (Electronic Health Records) too. The work in [37], introduces a searchable privacy-preserving enciphering mechanism for encrypted personal health records stored in MongoDB, while the article in [38], proposes a system to manage the intensive analytic workloads by realizing them in a NoSQL data store, where possible, while code in the R language is used to perform any procedural computations. Finally, the authors in [39], present an AQL (Archetype Query Language) interpreter operating on top of the MongoDB NoSQL query language, to efficiently perform storage and retrieval operations in EHRs.

To the best of our knowledge, the present work constitutes the first system that supports the collection, management, analysis, and sharing diverse, multi-faceted data in the CH and tourism domains, allowing users without an IT background to deploy, populate, and manage their own data ponds within minutes, alleviating the need to rely on expensive custom-made solutions that require IT infrastructure and skills to maintain.

III. System Architecture

Inspired from our previous work [40] and taking into account the experience drawn from the literature surveyed in the previous section, we reconstructed the Hydria data lake [40] applying cutting-edge NoSQL technologies, in order to enhance the system's performance and scalability. In this chapter, we mainly emphasize on the Data Storage and Management unit, while we briefly outline the other components of the Hydria data lake system. Broadly, the proposed data lake has the ability to (i) gather/import structured, semi-structured and unstructured data out of varying digital sources, (ii) pile up user-generated survey records through appropriately crafted questionnaires, (iii) maintain, handle and arrange the captured data in separate data ponds (tailored data collections used to conceptually group data within an individual CI/tourism application), (iv) distribute entire data collections or data collection portions to other users bearing similar interests through a robust Pub/Sub sharing tool, (v) explore, filter and examine stored data employing a powerful, yet easy-to-use data visualization widget, which executes dynamic queries in the background, with a view to present various graphical representations of information, and (vi) administer system users, defining and fine-tuning access rights on the data. Figure 1 depicts a high-level perspective of the system's architecture, the infrastructure's distinct operational tiers, their functionality and their arrangement and interoperation within the data lake framework.

A. Data Storage and Management trier

The Data Storage and Management (DSM) unit is responsible for manipulating the data collected and/or created by the Data Harvesting module and the Input Collection Manager respectively, and maintaining them in data ponds. The DSM unit additionally supports agile and versatile methods for designing, managing and broadening a data pond (or a data pond template). In the system's core lies MongoDB database, which underpins the construction, modification, arrangement and administration functionalities of each data pond, while it also undertakes the four basic CRUD (Create, Read, Update, and Delete) operations on persistent storage, i.e., it manages

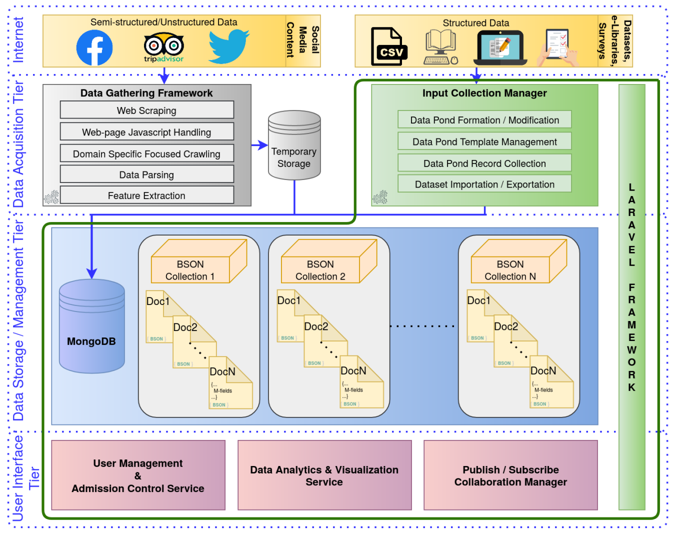

Internet
Semi-structured/Unstructured Data
Social Media Content
Structured Data
CSV
Datasets, e-Libraries, Surveys
Data Acquisition Tier
Data Gathering Framework
Web Scraping
Web-page Javascript Handling
Domain Specific Focused Crawling
Data Parsing
Feature Extraction
Temporary Storage
Input Collection Manager
Data Pond Formation / Modification
Data Pond Template Management
Data Pond Record Collection
Dataset Importation / Exportation
LARAVEL FRAMEWORK
Data Storage / Management Tier
MongoDB
BSON Collection 1
BSON Collection 2
BSON Collection N
Doc1
Doc2
DocN
M-fields
BSON
User Interface Tier
User Management & Admission Control Service
Data Analytics & Visualization Service
Publish / Subscribe Collaboration Manager

Fig. 1. System architecture separated in four layers (starting from the upper to the lower level): i) the internet layer illustrates the heterogeneous data sources on the web, ii) the second layer depicts the data harvesting mechanisms, iii) the data storage and management layer constitutes the system's back-end and iv) the UI level presents the various services offered to system users.

the stockpiled data associated with a specific data pond at the physical level.

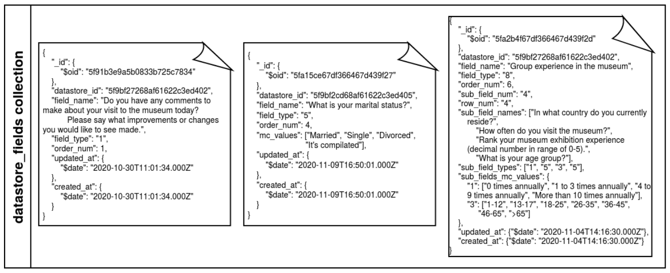
According to the MongoDB terminology [41], a MongoDB database constitutes a physical repository for *collections*. A *collection* is an equivalent notion to a table in an RDBMS; a Collection represents a set of Documents following the *BSON syntax*, which is similar to the JSON syntax, although BSON is an extended version of JSON format implemented by MongoDB. A MongoDB Document can be considered as the counterpart of a tuple (or row) in an RDBMS. Additionally, a Document comprises a set of fields, where each field is a name/value pair, similar to the concept of column of an RDBMS. With respect to document indexing, MongoDB spontaneously indexes all documents based on the `_id` key. The `_id` key ensures a unique identification of each document in a specific collection and can be correlated to the Primary Key concept of an RDBMS. Moreover, MongoDB can create indexes on embedded documents, or even create composite indexes combining two or more keys to compose a particular index. Taking advantage of embedded documents and linking, MongoDB can easily relate records avoiding table joinning operations employed in SQL philosophy, allowing thus for considerable performance improvements, since table joining is a costly operation. Finally, MongoDB provides the aggregation pipeline framework, where several processing stages are organized into a pipeline, to efficiently compute the desired outcome. This aggregation framework uses the MapReduce robust mechanism in the background of MongoDB; the aggregation pipeline framework bears analogies to the aggregate functions (e.g., count(), sum(), etc.) supported by SQL in RDBMSs, where the values of multiple rows are grouped together as input on certain criteria to produce the requested result. Table I summarizes the aforementioned MongoDB terms, correlating

TABLE I
MONGODB CONCEPTS CORRELATED WITH THE CORRESPONDING RDBMS CONCEPTS.

MongoDB Terms	RDBMS Terms
Database	Database
Collection	Table
BSON document	Row (or tuple)
Field	Column
Index	Index
Embedded documents and linking	Table joins
`_id` field	Primary key
Aggregation pipeline framework	Aggregation (e.g., Group By)

them to the respective terms in RDBMS terminology.

1) Data pond management: The proposed system offers various services assisting its users with straightforward, intuitive point-and-click methods to create and edit the desired data ponds. The creation process comprises determining a unique title and providing a description for the new data pond; subsequently, the data pond is stored into the system catalog for `datastores_index` collection; thereupon, the user is able to edit the newly created data pond defining several data fields by specifying a textual passage as the field name, and one of the available data types as the field type. Field types, besides being used for data type checking, are also utilized in the questionnaire construction process, where data is directly sourced from users via dynamically constructed forms. Every constructed type-independent data field of every data pond is deposited in the `datastore_fields` collection, which effectively realizes a system dictionary for data fields. For example, as we can observe in figure 2, the document on the left outlines a simple text type field containing just the field's textual description (`field_name`), its answering type (`field_type`) and some meta-data elements (such as the field's unique id, its order in the data pond and its creation and modification timestamps). The definition of multiple choice-type fields requires that the above-presented field specification document is complemented with an additional field (`mc_values`), which holds an array of acceptable values that can be used for this particular multiple choice field. Complex data type fields can be also be defined, consisting of equivalent content with the documents already stated, however, it additionally accomodates the number of times that this complex data type field will occur (`row_num`) in the data pond and an embedded document (`sub_fields_mc_values`), which, in turn, encapsulates two arrays of multiple choice response values and the number (starting from zero) of the respective sub-field name that the array belongs to. Complex data fields are described in detail in subsection III-A2. All three document types are stored in the system dictionary for data fields (i.e. the `datastore_fields` collection). In figure 2, we can observe that documents on the left and on the right share the same `datastore_id` value, which means that both documents appear in the same data pond. It is worth mentioning that based on the selected answering data


datastore_fields collection

```
{
    "_id": {
        "$oid": "5f91b3e9a5b0833b725c7834"
    },
    "datastore_id": "5f9bf27268af61622c3ed402",
    "field_name": "Do you have any comments to
make about your visit to the museum today?
        Please say what improvements or changes
you would like to see made.",
    "field_type": "1",
    "order_num": 1,
    "updated_at": {
        "$date": "2020-10-30T11:01:34.000Z"
    },
    "created_at": {
        "$date": "2020-10-30T11:01:34.000Z"
    }
}
```

```
{
    "_id": {
        "$oid": "5fa15ce67df366467d439f27"
    },
    "datastore_id": "5f9bf2cd68af61622c3ed405",
    "field_name": "What is your marital status?",
    "field_type": "5",
    "order_num": 4,
    "mc_values": ["Married", "Single", "Divorced",
                  "It's compilated"],
    "updated_at": {
        "$date": "2020-11-09T16:50:01.000Z"
    },
    "created_at": {
        "$date": "2020-11-09T16:50:01.000Z"
    }
}
```

```
{
    "_id": {
        "$oid": "5fa2b4f67df366467d439f2d"
    },
    "datastore_id": "5f9bf27268af61622c3ed402",
    "field_name": "Group experience in the museum",
    "field_type": "8",
    "order_num": 6,
    "sub_field_num": "4",
    "row_num": "4",
    "sub_field_names": ["In what country do you currently
        reside?",
            "How often do you visit the museum?",
            "Rank your museum exhibition experience
        (decimal number in range of 0-5).",
            "What is your age group?"],
    "sub_field_types": ["1", "5", "3", "5"],
    "sub_fields_mc_values": {
        "1": ["0 times annually", "1 to 3 times annually", "4 to
        9 times annually", "More than 10 times annually"],
        "3": ["1-12", "13-17", "18-25", "26-35", "36-45",
            "46-65", ">65"]
    },
    "updated_at": {"$date": "2020-11-04T14:16:30.000Z"},
    "created_at": {"$date": "2020-11-04T14:16:30.000Z"}
}
```

Fig. 2. Example of BSON documents representing three data pond fields. The document on the left denotes a field of a text-type answer, the document in the middle shows a multiple choice-type field, while the document on the right defines a complex type response field. All three documents are of different structure and contain dissimilar content, however all of them are stored within the `datastore_fields` collection.

type, hidden panels or dialogs occur prompting the user to enter proper material for the particular element (e.g., if the chosen answering data type is of multiple choice category, the user has to be presented with a list of appropriate values or has to pick one of the available lists already stored in the `multiple_choice_lists` collection). The available data types that are currently supported by the proposed solution are as follows: title (non-fillable field acting as a label to separate data pond field sections), text, integer, decimal, date, multiple choice, picture drawing, image file, and complex data types.

2) Complex data types: Complex data types are composite and advanced data types that are made available to empower the data pond design process, providing a methodical and effective way of modeling groups of fields that occur more than once within a data pond record, among separate records of the same data pond, or even within records of distinct data ponds. The document on the right side of figure 2 illustrates an example of a complex data type field, as stored in the MongoDB database collection. The composite data type defined in this example models the museum experience of a group of four people (e.g., a family of four), comprising four sub-fields (`sub_field_names`) of different data types (`sub_field_types`). These sub-fields appear four times and each of these appearances corresponds to an individual occurrence of the field group; therefore, the four occurrences allow for the accommodation of four distinct replies in a questionnaire. A user may append a complex data type field on a data pond, providing the specification of a recurring element containing multiple fields. It should be pointed out that a complex data type definition may be restructured at a later time by editing, deleting and/or rearranging any individual sub-field via an appropriate pop-up wizard; any modifications on a complex data type are reflected within the relative data pond. The creation and usage of complex data types offer: (i) higher versatility in the formulation process of a data pond, (ii) enhanced modeling of the input data, facilitating information acquisition, storage and management and thus, (iii) more eloquent and semantically richer query possibilities.

3) Reusing and sharing data shards: To guarantee information consistency across data ponds, and improve data integrity and validation of input, the proposed system provides users with the ability to:

1) compose and share data pond templates by promoting the reuse of all or collections of data pond fields (e.g., a survey of demographic information) among several data ponds. To advocate template usage, an integrated mechanism prompts users to consider using any of the already stored templates once she edits a new data pond for the first time. Constructing and editing a template is roughly the same procedure as generating a data pond, however, information about templates is stored in the `templates_index` collection.
2) dynamically create, store, and edit drop-down lists of elements. Creating a new drop-down list requires defining a unique list label and specifying the list elements. The drop-down list is then stored in the `multiple_choice_lists` collection and can be imported in any data pond containing a multiple choice field that conceptually corresponds to this list.

4) Populating data ponds: Once a data pond has been created, a user is able to populate it with data, employing any of the available data acquisition services (described in detail in III-B1) provided in this data lake context. For instance, the following population methods may be used: (i) initiation of automated data gathering crawlers, through the *Data Gathering Framework*, which is capable of navigating the web and popular social media platforms, identifying, gathering and stockpiling within a data pond content of interest; (ii) by employing the *Input Collection Manager* that provides: (a) the construction of survey-style digital forms authorizing users to manage data collection operations that concern electronic data input of end-users into structured forms, (e.g., surveys, end-user evaluations, museum experience records, etc.), and (b) the automatic loading of CSV/XML/JSON formatted datasets. At the system back-end, the MongoDB schemaless database offers an elastic and flexible way to store data in its collections. Practically, before storing the first document within an individual data pond, the specific data pond is not physically realized at storage level: only a description of the fields that it would contain is stored in the system dictionaries. Subsequently, when the first document/record is inserted, a new collection and its fields (name-value pairs) are dynamically generated within MongoDB database; the collection name inherits the unique data pond title stored in the `datastores_index` collection, while each document field name acquires the field name (which has been stored in the `datastore_fields` collection) that corresponds to this particular data pond through the `datastore_id` field; additionally, the field is assigned a value from the collected content that matches its data type. The advantages of applying MongoDB in the back-end of our approach include intuitive, adaptable and versatile data pond management, a straightforward way to scale up already stored and loaded data ponds (e.g., after completing the data population process, a user is able to redefine a particular data pond by adding/editing/removing any number of data pond fields), as well as fast performance and good scalability, with query execution time remaining practically unaltered as workload increases.

B. Further data lake components

In this subsection, the rest of the components showin in figure 1, are described.

1) Data acquisition tier: Browsing the web, including tourism-related sites (e.g. the Odysseus platform of the Greek Ministry of Culture[1]), online encyclopedias, digital libraries, portals and cultural govermental websites, large amounts of information of interest in the CH domain and in the tourism sector can be retrieved. Besides that, nowdays, large volumes of data concerning the CH field and the tourism sector are produced and uploaded in popular SM platforms like Facebook, TripAdvisor or Twitter. This information can be utilized in numerous cultural and tourism-related applications, and therefore the capturing of this information constitutes an essential task for many cultural and tourism-related foundations. To support this need for information capturing in the domains of culture and tourism, the proposed approach offers an efficient data acquisition unit, that comprises two individual modules as illustrated in the *Data Acquisition Tier* of figure 1.

- The *Data Gathering Framework* provides various web scraping services allowing users to configure and launch automated data acquisition tasks against widespread SM platforms (currently Facebook and TripAdvisor crawlers are available, while support for more SM platforms is under development). At the heart of this module lies the open source crawling framework Scrapy[2], which is responsible for scraping SM content. To initiate the crawling mechanism, the appropriate web scraping spider needs to be provided with the initial seed URLs; subsequently, the spider identifies the respective SM platform, navigates in the HTML elements of the current page, recognizes and extracts the targeted data (e.g., cultural venues, PoIs, reviews). However, in their efforts to make their content more interactive and improve user experience, SM platforms utilize to a great extent Javascript, Ajax and dynamic content, making the crawling process and data extraction process more tricky; to tackle this issue, the Data Gathering Framework utilizes the Selenium library[3], which provides a web browsing method that simulates human behaviour through a browser on a given website and allows the programmatic collection of web page data. Furthermore, the Data Gathering Framework involves a focused crawl service, utilizing the ACHE crawling infrastructure[4], tailored to operate thematic crawls on the open web with the view to uncover new resources that might hold information of interest in the CH field and the tourism domain. The ACHE crawler ranks URLs in the crawl frontier and classifies the crawled pages as relevant or irrelevant using machine learning techniques. It has to be noted that the Data Gathering Framework includes provisions to respect user privacy and ensure -to the greatest extent possible using reasonable means- that no individual may be identifiable through the collected data.
- The *Input Collection Manager* supports a variety of services supplying users with suitable mechanisms to (i) manage (in a stand-alone data pond) questionnaire-style digital forms tailored for cultural surveys, as well as (ii) gather the associated survey electronic records filled up by end-users, who have access on the particular data pond (described in detail in *User Management and Admission Control* service in III-B2); (iii) reuse entire or segments of the questionnaires created through the *Data Pond Template Management* service; and (iv) import/load structured CSV/XML/JSON formatted datasets, via a robust document importing mechanism, that automatically matches the columns of the file with the pre-defined data pond fields and for each data item (commonly a row in the CSV file or an element under the root of the XML/JSON document) a new data pond record is created and stored. Additionally, we have to note that all of the above functionalities are native in our work and are linked with the *Data Storage and Management* service (described in detail in III-A).

2) User interface tier: As depicted in figure 1, the User Interface layer consists of three modules related with the system's *User Management and Admission Control*, the *Data Analysis and Visualization* and the *Publish/Subscribe Collaboration* services respectively.

- Regarding the *User Management and Admission Control*, this module is responsible for administering the system's users by granting the appropriate rights and permissions, defining thus access control on data ponds and the records stockpiled within them. The users in the system's community are classified in the following categories: (i) *System administrators*, who are granted with all privileges and are capable to manage all data ponds and the collected records, while they are allowed to administer all user types and assign user roles in the data lake; (ii) *Power-users* are broadly curators responsible for their own data ponds (and the data stored within them). They are supplied with the suitable permissions to compose and manage new data ponds, establish data gathering procedures (c.f. section III-B1); they are able to access, filter, investigate and visualize collected data, while they may request from the system administrator to attach certain end-users in the data ponds they possess, or even cooperate with other power-users, who bear similar interests and wish to participate in a specific study by applying the Publish/Subscribe tool; (iii) *End-users* are located at the lowest level of the user management hierarchy, as they are provided with fewer functionalities and capabilities in the data lake ecosystem. The are able to contribute

[1]http://odysseus.culture.gr/
[2]https://docs.scrapy.org/en/latest/intro/overview.html
[3]https://pypi.org/project/selenium/
[4]https://github.com/VIDA-NYU/ache

in data collection tasks by creating/viewing/editing electronic records out of questionnaire-style forms in the data ponds they are assigned to; they are allowed to apply shallow analysis operations on their own contributed data; however, they are neither able to manage data ponds, nor view records created by other users across the same data pond.

- The *Data Analysis and Visualization* service provides a robust, point-and-click query and data handling tool allowing users to explore, filter out and analyze multiple fields of the stored data of each data pond, without necessitating any MongoQL (MongoDB Query Language) knowledge or any form of IT experience. Additionally, this service supports data visualization by exporting charts in miscellaneous forms such as histograms, pie charts, (heat) maps, (stacked) bars/columns, area/mekko/bubble charts and scatter plots. Exporting a graph requires a procedure of three phases where the user: (i) determines the chart type, (ii) selects the desired dataset by designating a data pond and any number of its fields, and (iii) establishes filtering conditions/restriction on the specified dataset (if needed). It is worth mentioning that the data lake environment assists users by offering them online assistance with examples for the various data analysis elements.
- Regarding the *Publish/Subscribe Collaboration* mechanism, this handy and user-friendly tool supports the sharing of datasets (or dataset segments), the exploration of the available data ponds, as well as the cooperation between power-users within the data lake environment. A power-user is able to use this functionality by applying the subsequent two-step procedure: (i) initially, she has to search for already stored data ponds that fulfill the given keyword-based query; after picking out any of the resulting data ponds, she is able to register a subscription request to the possessor(s) of the chosen data pond(s), seeking access permission on the data pond's contents (schema definition). The data pond owner can accept or decline the subscription request; if the request is accepted, the initial power-user can access the data pond's contents; (ii) afterwards, having access in the data pond's schema, the power user may choose any number of data pond fields and send a follow-up subscription request to the data pond owner(s) to access the selected data pond fields at record level. In the same manner, the owner(s) of the data pond(s) may accept or deny the request as is, or opt to share only specific data pond field records, selectively granting thus access to a subset of the fields initially requested for.

Once both steps have been completed, the requesting power-user has obtained access to view the data to which she has subscribed, employ them alongside with her own data ponds and/or export visualization graphs. Any newly introduced records matching the subscription are incorporated in the subscribed data pond, while the subscribed power-user will be appropriately notified.

3) Implementation overview: Our data lake environment has been entirely developed using open source software. The Data Gathering Framework (shown in Data Acquisition Tier of figure 1) is implemented exploiting the LAMP (Linux/Apache/MariaDB/PHP) solution stack for temporary storage of the extracted data and was developed using Python tools. The other built-in components were developed applying the web application framework Laravel and the NoSQL DBMS MongoDB as the back-end of our system's infrastucture (in figure 1, these components are surrounded by the solid green line). To enhance system interactivity, many operations utilize the JavaScript/JQuery/AJAX programming languages.

IV. Conclusion and Future Work

In this paper we have presented a novel, integrated system for collecting, managing, analyzing and sharing diverse, multi-faceted CH/tourism-related data. The proposed system can gather data from both structured and unstructured/semi-structured sources, and stores all data under a homogenized scheme in a flexible, document-oriented store. Users may access the data either on-request, through data analytics and visualization services, or according to a publish-subscribe scheme. The proposed system also includes functionality for administrators to manage users and access rights to the content. The system is designed to require low or no IT expertise for deploying, populating and managing data collections, facilitating and accelerating the relevant operations.

In our future work, we aim to conduct extended performance evaluation experiments to compare in detail the performance cultural/tourism-related data lakes employing NoSQL DBMS, such as MongoDB, and hybrid cultural/tourism-related data lakes realizing warehousing techniques over RDBMSs. We also plan to develop a module to supply the data lake environment with the ability to understand scripts utilizing the proper NLP (Natural Language Processing) tools, with the view to achieve sentiment analysis of the captured review texts and underpin recommendations on touristic destinations and points-of-interest for users.

V. Acknowledgment

This research has been co-financed by European Union and Greek national funds through the Operational Programme "Competitiveness, Entrepreneurship and Innovation", under the call RESEARCH—CREATE—INNOVATE (project code: T1EDK - 03874)

References

[1] N. A. Ghani, S. Hamid, I. A. T. Hashem, and E. Ahmed, "Social media big data analytics: A survey," *Comput. Hum. Behav.*, vol. 101, pp. 417–428, 2019.

[2] S. B. Abkenar, M. H. Kashani, E. Mahdipour, and S. M. Jameii, "Big data analytics meets social media: A systematic review of techniques, open issues, and future directions," *Telematics Informatics*, vol. 57, p. 101517, 2021.

[3] H. Dabbèchi, N. Z. Haddar, H. Elghazel, and K. Haddar, "Nosql data lake: A big data source from social media," in *HIS, Virtual Event, India, December 14-16, 2020*, ser. Advances in Intelligent Systems and Computing, A. Abraham, T. Hanne, O. Castillo, N. Gandhi, T. N. Rios, and T. Hong, Eds., vol. 1375. Springer, 2020, pp. 93–102.

[4] L. D. Valle and R. S. Kenett, “Social media big data integration: A new approach based on calibration,” *Expert Syst. Appl.*, vol. 111, pp. 76–90, 2018.

[5] G. Ritzer, P. Dean, and N. Jurgenson, “The coming of age of the prosumer,” *American behavioral scientist*, vol. 56, no. 4, pp. 379–398, 2012.

[6] J. Pybus, “Social networks and cultural workers: Towards an archive for the prosumer,” *Journal of Cultural Economy*, vol. 6, no. 2, pp. 137–152, 2013.

[7] O. Ormandjieva, M. Omidbakhsh, and S. Trudel, “Measuring the 3v’s of big data: A rigorous approach,” in *Joint Proceedings of the 30th International Workshop on Software Measurement and the 15th International Conference on Software Process and Product Measurement (IWSM Mensura 2020), Mexico City, Mexico, October 29-30, 2020*, ser. CEUR Workshop Proceedings, A. Abran and Ö. Özcan-Top, Eds., vol. 2725. CEUR-WS.org, 2020.

[8] V. Poulopoulos, C. Vassilakis, M. Wallace, A. Antoniou, and G. Lepouras, “The effect of social media trending topics related to cultural venues’ content,” in *SMAP, Zaragoza, Spain, September 6-7, 2018*. IEEE, 2018, pp. 7–12.

[9] S. Bampatzia, A. Antoniou, G. Lepouras, C. Vassilakis, and M. Wallace, “Using social media to stimulate history reflection in cultural heritage,” in *SMAP, Thessaloniki, Greece, October 20-21, 2016*, I. Anagnostopoulos and I. Paraskakis, Eds. IEEE, 2016, pp. 89–92.

[10] C. Vassilakis, V. Poulopoulos, M. Wallace, A. Antoniou, and G. Lepouras, “Tripmentor project: Scope and challenges,” in *CI@SMAP, Larnaca, Cyprus, June 9, 2019*, ser. CEUR Workshop Proceedings, A. Antoniou and M. Wallace, Eds., vol. 2412. CEUR-WS.org, 2019.

[11] A. Alharthi, V. Krotov, and M. Bowman, “Addressing barriers to big data,” *Business Horizons*, vol. 60, no. 3, pp. 285–292, 2017.

[12] J. Han, E. Haihong, G. Le, and J. Du, “Survey on nosql database,” in *2011 6th international conference on pervasive computing and applications*. IEEE, 2011, pp. 363–366.

[13] A. Oussous, F. Benjelloun, A. A. Lahcen, and S. Belfkih, “Nosql databases for big data,” *Int. J. Big Data Intell.*, vol. 4, no. 3, pp. 171–185, 2017.

[14] A. Corbellini, C. Mateos, A. Zunino, D. Godoy, and S. N. Schiaffino, “Persisting big-data: The nosql landscape,” *Inf. Syst.*, vol. 63, pp. 1–23, 2017.

[15] K. Dehdouh, O. Boussaid, and F. Bentayeb, “Big data warehouse: Building columnar nosql OLAP cubes,” *Int. J. Decis. Support Syst. Technol.*, vol. 12, no. 1, pp. 1–24, 2020.

[16] R. Esbai, F. Elotmani, and F. Z. Belkadi, “Toward automatic generation of column-oriented nosql databases in big data context,” *Int. J. Online Biomed. Eng.*, vol. 15, no. 9, pp. 4–16, 2019.

[17] B. Imasheva, A. Nakispekov, A. Sidelkovskaya, and A. Sidelkovskiy, “The practice of moving to big data on the case of the nosql database, clickhouse,” in *Optimization of Complex Systems: Theory, Models, Algorithms and Applications, WCGO, Metz, France, 8-10 July, 2019*, ser. Advances in Intelligent Systems and Computing, H. A. L. Thi, H. M. Le, and T. P. Dinh, Eds., vol. 991. Springer, 2019, pp. 820–828.

[18] P. Suárez-Otero, M. J. Mior, M. José Suárez-Cabal, and J. Tuya, “Maintaining nosql database quality during conceptual model evolution,” in *2020 IEEE International Conference on Big Data (Big Data)*, 2020, pp. 2043–2048.

[19] A. Khalil and M. Belaïssaoui, “New approach for implementing big datamart using nosql key-value stores,” in *5th International Conference on Cloud Computing and Artificial Intelligence: Technologies and Applications, CloudTech 2020, Marrakesh, Morocco, November 24-26, 2020*, M. Essaaidi, M. Zbakh, and A. Ouacha, Eds. IEEE, 2020, pp. 1–6.

[20] R. Liu, H. Isah, and F. Zulkernine, “A big data lake for multilevel streaming analytics,” in *IBDAP*. IEEE, 2020, pp. 1–6.

[21] R. Li, H. He, R. Wang, S. Ruan, Y. Sui, J. Bao, and Y. Zheng, “Trajmesa: A distributed nosql storage engine for big trajectory data,” in *36th IEEEICDE, Dallas, TX, USA, April 20-24, 2020*. IEEE, 2020, pp. 2002–2005.

[22] S. Kim and Y. S. Kanwar, “Geoycsb: A benchmark framework for the performance and scalability evaluation of nosql databases for geospatial workloads,” in *2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, December 9-12, 2019*, C. Baru, J. Huan, L. Khan, X. Hu, R. Ak, Y. Tian, R. S. Barga, C. Zaniolo, K. Lee, and Y. F. Ye, Eds. IEEE, 2019, pp. 3666–3675.

[23] H. Mallek, F. Ghozzi, O. Teste, and F. Gargouri, “Bigdimetl with nosql database,” in *Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 22nd International Conference KES-2018, Belgrade, Serbia, 3-5 September 2018*, ser. Procedia Computer Science, R. J. Howlett, L. C. Jain, Z. Popovic, D. B. Popovic, S. N. Vukosavic, C. Toro, and Y. Hicks, Eds., vol. 126. Elsevier, 2018, pp. 798–807.

[24] A. A. Imam, S. B. Basri, R. Ahmad, J. Watada, and M. T. González-Aparicio, “Automatic schema suggestion model for nosql document-stores databases,” *J. Big Data*, vol. 5, p. 46, 2018.

[25] R. Sellami and B. Defude, “Complex queries optimization and evaluation over relational and nosql data stores in cloud environments,” *IEEE Trans. Big Data*, vol. 4, no. 2, pp. 217–230, 2018.

[26] M. H. Gharanai, R. S. Gh, and A. A. Rashid, “In the digital future: Revitalizing information management systems in afghan settings through not only SQL (mongodb) technology,” in *SKIMA, Chengdu, China, December 15-17, 2016*. IEEE, 2016, pp. 45–48.

[27] H. Abbes and F. Gargouri, “Big data integration: A mongodb database and modular ontologies based approach,” in *Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 20th International Conference KES-2016, York, UK, 5-7 September 2016*, ser. Procedia Computer Science, R. J. Howlett, L. C. Jain, B. Gabrys, C. Toro, and C. P. Lim, Eds., vol. 96. Elsevier, 2016, pp. 446–455.

[28] ——, “Mongodb-based modular ontology building for big data integration,” *J. Data Semant.*, vol. 7, no. 1, pp. 1–27, 2018.

[29] L. Karim, A. Boulmakoul, and A. Lbath, “Real time analytics of urban congestion trajectories on hadoop-mongodb cloud ecosystem,” in *Proceedings of the Second International Conference on Internet of things and Cloud Computing, ICC 2017, Cambridge, United Kingdom, March 22-23, 2017*, H. Hamdan, D. E. Boubiche, H. Toral-Cruz, S. Akleylek, and H. Mcheick, Eds. ACM, 2017, pp. 29:1–29:11.

[30] N. Q. Mehmood, R. Culmone, and L. Mostarda, “Modeling temporal aspects of sensor data for mongodb nosql database,” *J. Big Data*, vol. 4, p. 8, 2017.

[31] S. Hiriyannaiah, G. M. Siddesh, P. Anoop, and K. G. Srinivasa, “Semi-structured data analysis and visualisation using nosql,” *Int. J. Big Data Intell.*, vol. 5, no. 3, pp. 133–142, 2018.

[32] W. Jiang, L. Zhang, X. Liao, H. Jin, and Y. Peng, “A novel clustered mongodb-based storage system for unstructured data with high availability,” *Computing*, vol. 96, no. 6, pp. 455–478, 2014.

[33] G. Xu, Y. Ren, H. Li, D. Liu, Y. Dai, and K. Yang, “Cryptmdb: A practical encrypted mongodb over big data,” in *IEEE International Conference on Communications, ICC 2017, Paris, France, May 21-25, 2017*. IEEE, 2017, pp. 1–6.

[34] T. Jia, X. Zhao, Z. Wang, D. Gong, and G. Ding, “Model transformation and data migration from relational database to mongodb,” in *2016 IEEE International Congress on Big Data, San Francisco, CA, USA, June 27 - July 2, 2016*, C. Pu, G. C. Fox, and E. Damiani, Eds. IEEE Computer Society, 2016, pp. 60–67.

[35] M. P. Stevic, B. Milosavljevic, and B. R. Perisic, “Enhancing the management of unstructured data in e-learning systems using mongodb,” *Program*, vol. 49, no. 1, pp. 91–114, 2015.

[36] A. Rahartomo, R. F. Aji, and Y. Ruldeviyani, “The application of big data using mongodb: Case study with scele fasilkom UI forum data,” in *IWBIS, Jakarta, Indonesia, October 18-19, 2016*. IEEE, 2016, pp. 51–56.

[37] L. Chen, N. Zhang, H. Sun, C. Chang, S. Yu, and K. R. Choo, “Secure search for encrypted personal health records from big data nosql databases in cloud,” *Computing*, vol. 102, no. 6, pp. 1521–1545, 2020.

[38] S. Saini, S. P. Singh, and R. Agarwal, “Healthcare analytics with R and mongodb using social media,” *Int. J. Adv. Intell. Paradigms*, vol. 18, no. 4, pp. 552–567, 2021.

[39] M. R. Naveira, R. Sánchez-de-Madariaga, J. B. Castro, L. C. García, G. V. González, S. Pérez, M. P. Carrasco, F. Martín-Sánchez, and A. M. Carrero, “An archetype query language interpreter into mongodb: Managing nosql standardized electronic health record extracts systems,” *J. Biomed. Informatics*, vol. 101, p. 103339, 2020.

[40] K. Deligiannis, P. Raftopoulou, C. Tryfonopoulos, N. Platis, and C. Vassilakis, “Hydria: An online data lake for multi-faceted analytics in the cultural heritage domain,” *Big Data Cogn. Comput.*, vol. 4, no. 2, p. 7, 2020.

[41] D. Hows, P. Membrey, E. Plugge, and T. Hawkins, “Introduction to mongodb,” in *The Definitive Guide to MongoDB*. Springer, 2015, pp. 1–16.