Abstract— The transition from traditional to Big Data technologies has put new challenges in front of existing database modeling techniques. The “Design then implement” approach to relational databases modeling, could not be fully applied to “Discover and analyze” databases with rapid data growth. In the context of Big Data, models have preserved a descriptive role: they are used to analyze and understand the architecture of hybrid data structures. From the Model-Driven Engineering point of view, several questions emerge: Could Big Data and hybrid database models represent a formal specification for a solution, which can be transformed to an executable specification for a specific target platform, using automated transformations? Does the refining mechanism have its place in the process of hybrid data structure development? What is the role of modeling languages in the process of hybrid data structure specification? An approach to modeling hybrid databases, based on empirically validated concepts of Model-Driven Engineering, is presented in this paper. The approach was applied during the implementation of Precision Agriculture Based Information Service (PAIS) application – a web-based service for the collection and archiving of diverse data on crops, and their presentation to farmers.

I. INTRODUCTION

The mass usage of smart phones and applications, expansion of social networks and the emerging Internet-of-Things (IoT) [1] technologies have opened the door to a new technological shift of our society. This created new challenges for the data management area, with requirements to handle data with volumes expressed in petabytes. As a result, the concept of Big Data [2,3] emerged. This concept can be characterized as “4 V”: Volume (large data amounts), Velocity (large data change frequency), Variety (storage of data in different formats) and Value (valuable insights gained from the ability to analyze and discover new patterns and trends from high-volume and/or cross-platform systems).

Traditional database management systems [4,5] could not successfully handle the collection, storage, search and analysis of this type and amount of data. The complementary use of NoSQL (“not only SQL”) [6,7] databases (MongoDB [8], Cassandra [9], etc.) and distributed computing systems, such as Hadoop [10], represents one way to face the Big Data challenge. Hybrid databases [11], containing both relational and NoSQL characteristics, represent an alternative path for the implementation of the Big Data paradigm.

The advantages for combining SQL characteristics and NoSQL databases have been recognized by traditional relational database management system vendors. Microsoft, for instance, provided support for column storage in MSSQL Server 2012 [12]. On the other hand, projects such as Hive [13] or CQL [14] have become an SQL-like facade for NoSQL databases. The leader in Big Data application, Google, developed Megastore [15], a hybrid distributed database.

The transition from traditional to Big Data technologies brought challenges to existing techniques for data structure modeling. The “Design then implement, top-down” approach for modeling relational databases, with a predictable volume of information, could not be applied to “Discover and analyze” collaborative, interactive, non-relational databases, with exponential and rapid data volume growth.

Models [16], in the context of hybrid structures, did not completely lose the position they had in the process of traditional database development. ERwin® Data modeling tool [17] supports embedding NoSQL schemas into submodels of relational models, or creating mappings between relational and NoSQL databases, which provides a basis for migrating the data structure design. In the Big Data environment, models currently have a descriptive role: they are used to analyze and understand the architecture of hybrid data structures.

On the other hand, Model-Driven Engineering (MDE) [18,19,20] represents a methodological approach to software development, in which models represent a formal specification of the solution, which is then converted into an executable specification using automated transformations. From the MDE point of view, several questions emerge regarding the Big Data world: Can the refining mechanism [21,22] have its place in the process of hybrid data structure development? Do conceptual, logical and physical levels [23] exist, and what do they represent in Big Data technologies? What is the role of modeling languages [24,25] in the process of describing hybrid data structures? In this paper, the authors provide answers to some of these questions, with reference to results of developing hybrid data structure models using existing MDE concepts.

The remainder of the paper is structured as follows. The model driven relational database development process is described in Section 2. In Section 3, the methodology for the model driven hybrid database development is presented. An illustration of the application of this methodology using PAIS project as proof of concept is described in Section 4, whereas in Section 5 the conclusions and future steps are described.

1“This project is funded by FRACtALS (Future Internet Enabled Agricultural Applications, FP7 project No. 632874), under the funding framework of the European Commission”
II. MODEL DRIVEN DEVELOPMENT OF RELATIONAL DATABASES

Models have a central role in the domain of relational database design and implementation. Models represent a formal specification, which can then be transformed into a database schema using automatic transformations.

The model-centric approach to database development is based on concepts initially defined as part of the Model Driven Architecture (MDA) [26,27] methodology. MDA is a software design methodology, established by OMG consortium, and it represents a concretization of the MDE paradigm. The primary goals of the MDA approach (interoperability, transferability and reusability of code) are achieved by separating the specification of system functions from the concrete, platform specific implementation, as well as by standardizing the modeling languages for different usage domains. MDA is based on several basic building blocks, which represent OMG standards [28]: UML modeling language [29], MOF [30] as a constitutive core for the MDA approach, XMI [31] model serialization standard, three model types and multi-level transformations.

The MDA approach defines three basic model types (Fig.1) based on the abstraction level:

i) Computation Independent Model (CIM) – model with the highest abstraction level; the domain model, containing descriptions of concepts from the domain;

ii) Platform Independent Model (PIM) – describes the problem domain, while hiding concrete implementation technology details. It is therefore good for creating system specifications, leaving the technical details aside;

iii) Platform Specific Model (PSM) – model using concepts from the implementation platform. In most cases, the PSM is not created manually, but it is obtained using (semi-)automatic transformations on the PIM.

In MDA, the program code (or any other artifact) is the product of models, and it is generated using several transformations, which transform the model through different abstraction levels. The information is kept in the model for as long as possible, providing gradual model rafination. Raffinations [32] are model transformations, in which the source and destination model are on different abstraction layers. Raffination is often driven using additional data, which is manually included in the intermediate transformation products, Fig. 2. This process is also known as “model marking”, and it assures independency of the model from the concrete implementation platform.

Figure 1. Basic abstraction levels in MDA

Figure 2. Model marking [32]

The condition for using MDA approach in the traditional database development process is the choice of appropriate modeling languages, which enable the data structure to be specified in different levels of abstraction. It is established in practice [23] that the Conceptual Data Model (CDM), Logical Data Model (LDM) and Physical Data Model (PDM) are used in the process of relational database modeling (Fig.3). CDM, LDM and PDM correspond to CIM, PIM i PSM MDA levels respectively.

The CDM is used to describe concepts and relationships from the observed system. Entities from the conceptual model represent real-world concepts, and not database records. The CDM provides a means to analyze the conceptual information system structure, identify the entities which need to be represented in the model, their attributes and relationships between them. A CDM is more abstract than an LDM or PDM. The LDM is used to analyze the information system structure, independent from the specific physical database implementation. Although less abstract than the CDM, the LDM does not contain PDM specific objects (indexes, views etc.). The LDM can be used as an intermediate step between the conceptual and physical model in the process of database design. The PDM represents the structure of a concrete database. Entities from the LDM are transformed to tables in the PDM, and the relationships are transformed to foreign key constraints. An SQL DDL script can be generated from the PDM to create a database schema.

Figure 3. Data modelling layers


3 https://www.theregister.co.uk/2007/06/14/data_modelling_layers/
III. MODEL DRIVEN DEVELOPMENT OF HYBRID DATABASES

The following requirements led to the choice of model driven hybrid database development as a methodology for our solution:

i) one model needs to be used to describe an arbitrary number of schemas for different database management systems. The databases, for which schemas are integrated on model level, do not need to be physically integrated;

ii) it is possible to include/exclude the model of a new/existing database schema in/out of the existing model;

iii) the integrated model for the hybrid database schema can be partially transformed (different parts of the model can be used to generate different types of artifacts) and

iv) the approach used needs to be supported using existing (meta-) modeling tools.

Basic model driven relational database development concepts were generalized and applied in the context of designing hybrid databases. The three abstraction levels (CIM, PIM and PSM) were kept, whereas the models are transformed from one to the other using refinements. Due to the fact that the CIM level does not need to present specificities of different database management systems, the initial CDM was appropriate for modelling hybrid database structures on conceptual level. On the other hand, the PIM and PSM levels were used to describe informational and implementational characteristics of the data storages. This is why a controlled extension of the LDM and PDM needed to be implemented, in order to cover NoSQL database specific concepts.

Fig.4 illustrates part of the architecture for developing hybrid databases using model driven development approach:

i) specific database management systems were described using appropriate metamodels [34] (Document Metamodel, Key Value Metamodel, Graph Metamodel, RDBMS Metamodel etc.). Each metamodel contains specific characteristics of the selected system. The process of creating metamodels is illustrated using relationships with <<conceptualization>> stereotype. The metamodels can comply to any meta-metamodel (MOF, ECORE [35], KM3 [36]).

The integration of different metamodels was executed by transforming these metamodels into lightweight extensions [37,38,39] of the base metamodel. The metamodel of LDM and PDM was selected as the base metamodel. The process of transforming metamodels into lightweight extensions is illustrated using the relationship with <<transformation>> stereotype;

ii) according to the definition, lightweight extensions can be dynamically applied to, or removed from, the base metamodel. Using this approach it is possible to included new sub-models into existing models, with no information loss;

iii) using the origin of specific elements (relation with <<instanceOf>> stereotype) on model level (Hybrid Database Model on Fig. 4.), virtual sub-models were created (Graph DBMS SubModel, KeyValue DBMS SubModel, Document DBMS SubModel). Virtual sub-models represent inputs for dedicated generators, which output appropriate artifacts (relational database SQL schema, Hive queries for NoSQL DBs, etc.) and

iv) the usage of models, created by extending existing database modeling languages, has the characteristic of using existing technologies and user experience in the implementation of the MDE approach. Due to this fact, existing modeling tools, with support for lightweight extension mechanism, can be used to implement the described approach.

Figure 4. Architecture Model of Model-Driven Development of Hybrid Database
IV. APPLICATION

The described approach was used in the implementation of Precision Agriculture Based Information Service (PAIS), an application developed in 2015 and 2016. PAIS represents a web-based service for the collection and archiving of diverse data on crops, as well as their representation to farmers. It represents a complex service with heterogeneous data, consisting of aerial images and land sensor measurements, associated with spatial and temporal attributes. PAIS is an online service\(^4\), providing 24/7 access to the integrated image and sensor data from the parcels marked.

As part of the FRACTALS project, the development of PAIS required the use of FIWARE\(^5\) platform. FIWARE delivers a suite of generic enablers\(^6\), which allow the implementation of different functionalities. Due to the usage of FIWARE platform, the choice of using different databases in the implementation was made: sensor data are stored in Cosmos BigData\(^7\), drone images and image metadata are stored in the ObjectStorage\(^8\) (MongoDB implementation), whereas administrative data are stored in MySQL. A simplified architecture of PAIS system is illustrated in Fig. 5. The Data Storage Area represents the layer consisting of these databases. The choice of the database type to be used (MySQL, Cosmos or ObjectStorage) was made depending on the type and amount of data to be stored. Sensors provide large amounts of simple, unstructured data, so the Cosmos storage was selected. On the other hand, the data provided from the drones complied to the type of data the ObjectStorage was developed for (images with metadata).

Although Data Storage consists of three, physically separated, databases, during the modeling phase this layer was specified using a single model. The base metamodel used was the Physical Data Model. The model was extended using two lightweight extensions. One of the extensions represents MongoDB concepts, whereas the other one represents Cosmos BigData concepts.

Based on the relational database virtual sub-model, the logical and physical model, as well as SQL script for database table creation, were generated. The ObjectStorage virtual sub-model was used to generate REST services for reading data collected from drones, as well as for generating the flight preparation specifications, which were used to create the ObjectStorage structure. The BigData Cosmos virtual sub-model was used as input for the Hive query generator, used to read sensor data, as well as input for generators for new sensor registration specification. Based on these specifications, the data structure for BigData Cosmos storage was indirectly created.

V. CONCLUSION AND FUTURE WORK

In this paper, an approach to developing hybrid databases using the model driven approach is described. The approach is based on basic MDA concepts: multiple abstraction levels, usage of rafination and the existence of standard modeling languages used to create models on different abstraction levels. The differences between

---

\(^4\) http://www.agro-pais.com/#/about
\(^5\) https://www.fiware.org/
\(^6\) https://catalogue.fiware.org/enablers?page=1
\(^7\) https://catalogue.fiware.org/enablers/bigdata-analysis-cosmos
\(^8\) https://catalogue.fiware.org/enablers/object-storage-ge-fiware-implementation
concrete NoSQL databases and traditional databases systems were described using lightweight extensions of existing metamodels.

The model driven hybrid database development approach was used during the implementation of PAIS application. PAIS represents a complex service with heterogeneous data, consisting of aerial images and land sensor measurements, associated with spatial and temporal attributes. To suit the project needs, the logical and physical database model were extended, to provide support for a single NoSQL database type (Document-oriented databases [3]). As part of future work, the aim is to create extensions to include key-value and column-oriented databases.

REFERENCES