Smart City architecture for data ingestion and analytics: processes and solutions

Pierfrancesco Bellini, Paolo Nesi, Michela Paolucci, Imad Zaza
DISIT lab, University of Florence, Http://www.disit.org <name>.<surname>@unifi.it

ABSTRACT. Smart city architectures have to take into account a large number of requirements related to the large number of data, different sources, the need of reconciliating them in a unique model, the identification of relationships, and the enabling of data analytics processes. Ingested data, static and realtime, must be stored, aggregated and integrated to provide support for data analytics, dashboard, making decision, and thus for providing services for the city. This means: i) compatibility with multiple protocols; ii) handle open and private data; iii) work with IOT/sensors/internet of everything; iv) perform predictions, behavior analysis and develop decision support systems; v) use a set of dashboards to make a real-time monitoring of the city; vi) consider system's security aspects: robustness, scalability, modularity, interoperability, etc. This approach is determinant to: monitor the city status; connect the different events that occur in the smart city; provide support for public administrators, police department, civil protection, hospitals, etc., to put in action city/region strategies and guidelines and obviously directly to the citizens. In the paper, we focus on data ingestion and aggregation aspects, putting in evidence problems and solutions. The solution proposed has been developed and applied in the context of the Sii-Mobility national smart city project on mobility and transport integrated with services. Sii-Mobility is grounded on Km4City ontology and tools for smart city data aggregation and service production.

Keywords: Big Data, smart city data warehouse, Smart City Architecture/Platform, Smart City Ontology, Decision Support Systems.

1. INTRODUCTION

The main technical issues regarding smart city solutions are related to: data access, aggregation, reasoning, access and delivering services via Smart City APIs. The final aim is serving city users in a smarter and efficient manner. Therefore, collected and produced data are used to facilitate the creation of smart and effective services exploiting city data and information. This means to make effective and efficient the data access with their semantics, the service delivering, the access to define and control dashboards, and the interoperability with any other smart control systems active in the city (e.g., mobility, energy, telecommunication, fire brigade, security, etc.). In the world, municipalities/cities and public administrations are publishing huge amount of open data. These data can be coarsely aggregated for integration by using solutions such as CKAN [CKAN], OpenDataSoft [OpenDataSoft]. In some cases, they provide access to effective datasets, by using some data integration and visualization tools which provide the possibility of creating graphic charts, such as distributions or pies, on the basis of the values contained in the dataset. In sporadic cases, they also provide access to datasets as Linked Data (LD), Linked Open Data (LOD), coding data information in terms of RDF triples [Barneers et al.]. Very rarely, they can provide data from some RDF store endpoints to make SPARQL queries on the data exploiting some ontology and other entities [SPARQL], rather than working only on metadata. The access to RDF stores for data browsing can be performed by using visual browsers as in [Bellini et al., 2014]. In most cases, the effectiveness of data service system for Smart City is enabled by the availability of private data owned and managed by City Operators addressing specific domains: mobility operator, energy providers, business services (health, water), telecom operators, tourist operators, universities, etc. Real-time data are provided by city operators through some APIs as Web Services or REST calls. The APIs for providing data to the data aggregator of the city may be compliant with multiple standards (such as DATEX II for mobility, intelligent transport system for public services, parking; IETF [IETF], ETSI [Lin et al., 2013] or OneM2M [Swetina et al., 2014] for Internet of Things (IOT), Green Button Connect for energy data collection. Thus, the developers may collect data that still need to be aggregated to make them semantically uniform, referring to the same elements in the city, and to establish multiple agreements.

The effective deploy of smart services for city users is very frequently viable only by exploiting the semantic integration of data as: open data, private data and real-time data coming from administrations and different city operators. This implies specific processes of reconciliation and the adoption of unifying data models and ontologies as in Km4City [Bellini et al., 2014b]. The semantic aggregation of data coming from several domains is unfeasible without a common ontology, since data are produced by different institutions/companies, by using different formats and aims, different references to geographical elements, and different standards for naming and identification adopted in different moments. Thus, datasets are rarely semantically interoperable each other since have been produced in different time, by different systems, by different people, etc. In addition, they may present different licensing models: some of them can be open, while other may be private of some city operator that would not be interested to lose the ownership by releasing them into an unregulated environment, or could simply provide some restrictions (e.g., no commercial); see for example the data of car sharing companies that are typically private of the company. Well aggregated and re-conciliated data for the identification of services and locations (open and private) can be exploited by reasoning algorithms for enabling sophisticated service delivering. For example, by providing suggestions and hints on route planning, inter-modality routing, parking, hospital finding in the case of emergence, finding specific point of interests, setting predictions (for parking and traffic) and detecting anomalies for early warning. The data values (actual, predicted and/or detected) can be delivered to different operators and city users by some personal assistants on the basis of the user profile and role. For example, to provide information about what is or what would be around a current GPS position, the integration of geographic information and services is needed; while the integration of geo-localized services and the assessment of typical people flows may help the city in improving public services and transport, providing suggestions to the city users, and planning changes in the city [Castillo et al., 2014].

This paper presents the work performed on defining smart city architecture and assessing its performance, as developed in the context of Sii-Mobility smart city project. Sii-Mobility aims to provide innovative services for mobility operators and city users moving in the city and in the region, to provide solutions for sustainable mobility and transport systems. In the literature, there are several proposals for smart city architectures, but only few of them are really in place with a relevant range of distinct kinds of processes, such as addressed in this paper. Most of other projects are mainly centered on manage wireless sensors networks, IoT such as the FLEXMETER platform [Patti et al., 2016], the Barcelona smart City Architecture [Sinaeepourfard et al., 2016] and the ALAMANC EU project [Bonino et al., 2015]. In Sii-Mobility, specific smart algorithms for data aggregation, personal assistance; solutions for dynamically shaping restricted traffic zones; production of personalized suggestions to allow the city users movements, aiming at improving sustainable mobility, etc. have been developed. These requirements necessitated a deep analysis of the state of the art proposals, to identify and then develop a solution allowing performing reasoning and deduction on city data collected from city operators, as open data and private data, as static and real-time data, as multiple domain data for producing suggestions and stimulus to city users [SIIMOB-DE1-1]. In this context, the main goals of an innovative and suitable architecture have been: (i) the data ingestion and aggregation services to integrate different kinds of data creating a real knowledge base for the city (a sort of expert system with inference capability), (ii) the computational capabilities and process management in the backend and in the front-end, (iii) the formalization of the Smart City API by which all the web and mobile Apps, and dashboards may have access to the smart city knowledge and services [Badii et al., 2017]. The work presented in this paper has been performed in the context of the Sii-Mobility smart city project on mobility and transport aspects and integrated with city services in general (http://www.sii-mobility.org). Sii-Mobility project includes 24 industrial partners from industry and research, and it has been partially founded by the Italian Ministry of Research as a special national program on Smart City, MIUR SCN. Sii-Mobility is focused on providing innovative services in Tuscany, which is an area of 3.5 million of inhabitants, and it involves the Tuscany Region and several municipalities and local governs in the area for the experimentations, covering almost all the Tuscany region.

The paper is organized as follows: in Section 2 the Sii-Mobility Smart City General Architecture is described; Section 3 presents the details related to the Smart City Data Ingestion and Aggregation Layer; in Section 4 the description of the Sii-Mobility Storage is reported. Section5 contains the Conclusions of our work.

2. Overview of Sii-Mobility Smart City Architecture

In this section, the key features of the Sii-Mobility architecture for smart city management are depicted. The architecture is described in **Figure 1**, putting in evidence the major components/tiers of the architecture. In general, the architecture is multitier and provide support for big data collection, analytics and intelligence, exporting / providing a number of services. In the following sub paragraphs, the major components will be better described, while details and a deeper discussion is provided for the most challenging aspects as discussed in the sequel.

The Ingestion Layer collects, harvests and processes various kinds of datasets and data streams characterized by high heterogeneity. For this reason, a process of data analysis and transformation is needed to make them interoperable and reusable. The data are collected in the form of: i) Open Data and come from: municipalities, Tuscany region (Observatory of mobility), LAMMA weather agency, ARPAT environmental agency, Social Media, etc.; ii) Private Data (data with some restriction) coming for example from City/Regionals Operators or personal mobile phones and regarding: users' actions, mobility, energy, health, cultural heritage, services, tourism, wine and food services, education, wellness, environment, civil protection, weather forecast, etc.

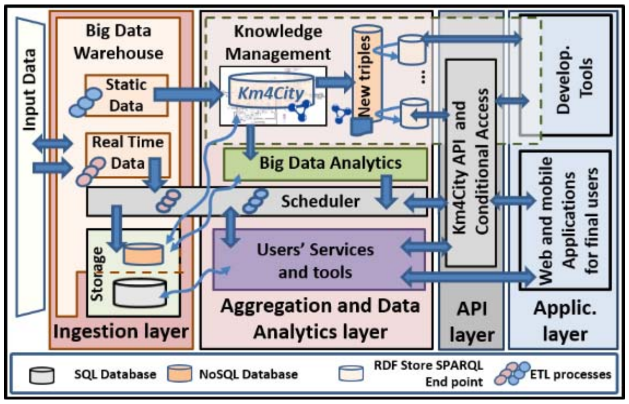

Figure 1. Sii-Mobility General Architecture.

Moreover, the data mainly comes from: (1) technically as web services, sensors, static files, etc., each of which, respecting a set of standards (or format types e.g. csv, json, html, xml, shape, etc.); (2) different providers: City Operators, Data Brokers, citizens, etc.; (3) different classification subareas (Point of interest, events, public Transports, traffic flows, etc.); and (4) can be both static and dynamic (or real-time); (5) Tv camera monitoring the territory; (6) social media crawled for collecting post related to the city; (7) city users from the their Apps and also from their specific contribution to participate to the city life, and with their profile, collecting profile for user kind (citizens, commuter, tourist, student, etc.); (8) IOT, internet of things sensors that may be provided by the operators as well as from the city users, etc.

The datasets are collected, improved in quality and successively aggregated and saved in the Sii-Mobility storage by using processes implemented as specific codes (as in the case of streams) or ETL. The latter mainly realized exploiting the Pentaho Kettle tool, for which a new module connecting Phoenix has been developed.

As a result, the Ingestion Layer elaborates data to save them into the central storage of the smart city architecture. Please note that, typically in a smart city a large number of entities described in with a number of attribute and/or producing data are present. So that, the best model for putting them in relationship is reticular to model the high number of relationships among entities. An example is the street graph. In addition, a relevant number of entities produce data over time. For example, real time data such as: traffic sensors, IOT devices, fuel prices, parking status, etc. The latter kind of data, are forming the historical data on which the data analytics and strategies are mainly performed. Thus, due to the large amount of data arriving in the storage and exploited by data analytics processes, a mixt of noSQL solutions has been adopted integrating graph database with tabular storages [HBase, Phoenix], [Virtuoso]. Thus, due to the complexity of the data flows, their variability, variety, and velocity we can be talk about big data architecture for smart city [Badii et al., 2017].

The **Aggregation and Data Analytics Layer** aggregates the data thanks to a Km4City multi-domain ontology creating a Knowledge Base for the city (a sort of expert system with inference capability) [Bellini et al., 2014b]. The creation of a unified model of relationships among the city entities in a common Knowledge Base if fundamental to enable effective data analytics and inferential processing related to the context of the city. The resulting processes and data are used for supporting decisions on the city strategies on the basis of: analysis, prediction, anomaly detection, early warning, suggestions, recommendations, etc. They can be used for improving the city services – e.g., intensifying the cleaning in specific areas, changing the schedule of the public transportation, changing the shape of the restricted traffic zone, tuning the price of parking for user profile and time slot, etc. Thus, the resulting data are made accessible on Dashboards, mobile App, notifications, reports, etc.

Therefore, the **Big Data Analytics algorithms** are capable to manage and produce new knowledge for the different kind of city users involved, such as citizens, students, commuter, tourists, city operators, public administrations, researchers and developers, etc. For example, the most active data analytics areas are devoted to, [Badii et al., 2017]:

- *data reconciliation* on the basis of a unified geographical references useful to make queries on the Knowledge Base along lines, area and for proximity; To this end a number of geolocation algorithms can be used, also processing the text with NLP approaches and exploiting the street graph into the knowledge base for recovering and connecting entities to geo locations as street, locality, region, POI, etc. Thus, allowing to provide answers to questions such as*: since I am on Bus 14 now, which is the bus stop at which I can found a very close milk shop for the baby.*
- *User Behaviors analysis* is typically performed on the basis of data collected from the Apps, TV camera, Wi-Fi access points, cellular networks of the telecom operators, registration to the hotels, accesses to the museums, usage of the public transportation, etc. The resulting elaborations in some cases can be clustered for user kind, age, nationality, sex, for the different time slot of the day, etc. The most appreciated resulting data are the list of most requested points of interests, the mostly adopted trajectories, the origin destination matrices, etc. [Bellini et al., 2017]
- *Predictions*: to provide a guess about how much the city services would be exploited the future, from the next minutes, to hours, weeks, etc. Predictions can be directly appreciated and used by city users, decision makers. For example predicting: available parking lots in any specific area, traffic flows and collapsing area, people flows, triage usage of hospitals, incidents in the streets, etc. Thus, answering to questions of the city users such as "How many available places there will be in the car park Firenze, tomorrow at 12:00?".
- *Early warning*: detecting anomalies in the city usage and behavior may help to identify the inception of unexpected events: a water bomb, a pipe broken, etc. Differently from the predictions, the identification of anomalies works comparing the current trends with respect to the predicted values, taking into account contextual data. For example, the arrival of a large amount of people in a square with respect to the typical values may be due to the organization of special events or to a closer disaster that is pushing the people to move. Typical data for the early warning detection can be obtained from environmental, traffic and people flow, water level, acoustic data, social media, etc.
- *Routing*: starting from the Knowledge Base and contextual information it is possible to provide answer to questions: *which is the fastest rout to reach the hospital for the ambulance taking into account busses, garbage collection, etc., what is the safest route to reach from A to B if I move to bike today, what is the most ecological (use of public transport, bike, on foot, etc.) route to get to Piazza Signoria passing by a baker?*. Most of the questions cannot be answered by commercial or state of the art applications such as Google Map, TomTom, Garmin, etc..
- *Suggestions and recommendations:* Suggestions and recommendations can be computed on the basis of: (i) the user profile collected from the App and from the social media, via OAuth, mutuated registration, or provided by the user directly; cumulated from the user behavior analysis; (ii) contextual information about the city from the knowledge base. For example, for providing advertising, or for suggesting alternative POI, etc.
- *User Assistant recommendations* can be computed on the basis of the contextual information and exploiting the user behavior and profile, for pushing them in taking a more virtuous behavior. For example, for stimulating the city users moving by their private cars to take the public transportation, or city users not walking enough to take a more healthy behavior. To this end, the behavior of the users has to be quite precisely monitored, for example, via specific Apps.
- *User Engagement requests* can be provided to the city users adopting the smart city APP to request them providing additional information, such as: number of people in a given area, to rank a service, to take a picture, to describe the queue, etc. The computation of the engagement has to be prepared in advance and exploit more or less the same data available for providing *User Assistant recommendations*. [Badii et al., 2017].
- *Ticketing and Booking:* it provides to the citizens a set of services regarding the purchase and reservation of tickets for events such as theatre, stadium, etc. or for public services such as the use of buses or tramway, may be providing a full travel ticketing with multimodal routing and multiple reservation from multiple providers.

Smart City API (Application Program Interfaces) Layer is fundamental to provide both: (i) aggregated data and (ii) services to Web and Mobile Applications, and eventually to third party city operators. Most of the Smart City APIs are substantially presenting a small set of features by using a REST Call service. In our case, the API are provided in multiple formats and modalities since Sii-Mobility is grounded on Km4City ontology and tools for smart city data aggregation and service production consist in the possibility of posing requests [Nesi et al., 2016]. The provided modalities for Smart City API are:

- **SPARQL Query** requests directly performed on the RDF Store endpoint using the standard SPARQL query protocol (based on REST) using GET or POST requests with the *query* parameter containing the SPARQL query, for example accessing to the Km4City endpoint;
- **SPARQL Query with Inference**: requests directly performed on the RDF Store endpoint http://servicemap.disit.org/WebAppGrafo/sparql, by using the SPARQL query protocol (based on REST) using GET or

POST requests with the *query* parameter containing the SPARQL query, including inference aspects in the case of Virtuoso, or automatically exploiting the inference in the case of OWLIM [Bellini et al., 2015b];

- **REST**: calls are performed by using APIs using full text, keywords, service ID (URI) to get info, geolocation, service ID (URI) to get closer services, time, etc.;
- **Query ID**: calls are performed by using a QueryID (identification) assigned by some visual tools to form and save the queries such as the ServiceMap (http://servicemap.disit.org) tool manager, after having performed a query by using the graphic user interface, as a visual query.

The Application Layer includes a set of applications that can be developed thanks to the Smart City APIs Km4City and can be divided in:

Development Tools: created to help the developers to find the different kind of data, data types, semantic information managed in the Sii-Mobility Knowledge Base:

- **Service Map** (http://servicemap.km4city.org) allows to visually formalize queries and generate calls compliant to Km4City Smart City API. These calls are directly sent via email to the developer for shortening the production of web and mobile applications as described in the sequel. [Badii et al., 2016]

- **Linked Open Graph** (LOG), which is a web application (http://log.disit.org) allowing the users to explore the Knowledge Base (KB) to see all the relations among the entities that are present in the KB also those hidden from the user interface. For example, it is possible to understand how the services are connected with the street graph or how to access to complete real-time information. The LOG allows to visually navigate the relations among the KB entities [Bellini et al., 2014].

- **SPARQL RDF Store Interface**, it allows the developers to pose SPARQL queries on the KB and get results.

- **Apache Zeppelin** which is a web-based notebook that enables interactive data analytics with dynamic visualizations on the distributed NoSQL storage.

Applications:

- **Mobile and web Applications**, such as the mobile App "Florence what where" (http://www.disit.org/app). Data about changes in the City Users status are collected continuously, and the system receives requests from the Mobile App periodically. Most of the services of the city are provided via the Apps, and on the other hand, the App can be used to collect user behavior, engagement, contributions, etc.

- **Dashboards for Control Rooms**, a set of dashboards monitoring high level view data. Typically, they are thematically created: mobility and transport, social media, energy, environment, health, resilience, etc. The Dashboard are designed to stay H24 on the wall of the control room or to be used by operators on their desktops. Dashboard Engine and Dashboard Builder, is a tool realized to automatize the creation of different dashboards. This because different final users can be interested in different information or can have access only to a limited group of data. Notificator, is a tool for generating and management of events with the following features: i) association of events generators, event types, messages, and recipients; ii) message book to define and manage messages; iii) address book to add and manage users and their e-mail addresses; iv) logging system for all the events monitored, to consult past events details with several search filters; v) Graphic User Interface (GUI) to add, edit or delete client applications to decide how to manage their notifications.

- **Participation tools**, a set of instruments to inform the city users about the city status. The information can substantially be provided to the city users via Web Pages on their computers via web as well as on specific Totem/Kiosks positioned in public locations (such as: station, bus stops, etc.). Belong to this type of tools also the (i) Variable Message Panels, VMP, which are typically positioned on the road entering in the city, or the bus stops, and that may be visualized limited information as text and very simple graphs; (ii) loudspeakers to inform the city users in the metro, in some square, on the Bus/train, (iii) direct call with automated calling system and SMS sending. These tools are typically exploited to advice the citizens and give them suggestion in case of critical events, such as: crowed condition, pollution condition, bomb attack, flooding of a river, fire, interrupted roads, etc.

- **ProcessLoader** allows at the non-technical users to upload, schedule and monitor processes in the Smart City back office. So that for uploading processes into the smart city back office, one does not need to know the functioning of the smart city complex tools. The processes can be ETL for data ingestion, as well as data analytics in multiple languages such as: Java, Python, R, C++. The Process Loader directly interfaces with the back-office process scheduler that is the DISCES (Distributed Smart City Engine Scheduler). The uploaded processes can be activated as periodic or sporadic/on demand execution. They may have a starting time or deadline, the typical duration, etc.

3. Managing Ingestion and Aggregation

In this section, a more detailed view on data ingestion and aggregation aspects and problems is presented. The data ingestion in the context of smart city implies the solution of a number of problems among them: (i) the management of several different data sets and streams, from static to dynamic data, coming from different sources, in different formats and with different rates that have to be integrated each other, to refer to the same city entities and services, (ii) the storage of ingested data to make them accessible from a big data storage for data analytics, referral and statistics data and just for showing historical trends in the different

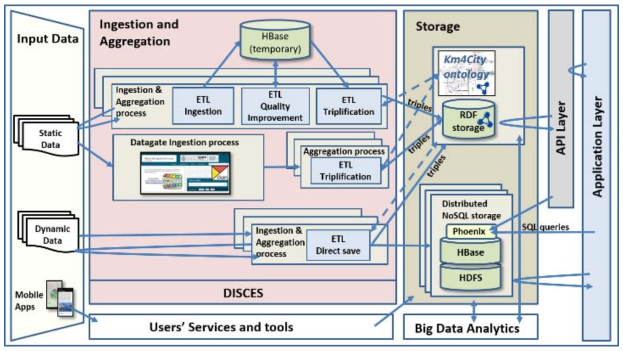

Figure 2. Sii-Mobility Detailed Architecture.

contexts.

The resulting detailed architecture is reported in Figure 2 where two main blocks are reported for describing the Ingestion and Aggregation and the Storage. Their design and functionality will be described in details in the next subsections.

3.1 Ingestion and Aggregation

The data to be ingested into the smart city can be Static or Dynamic. Static data are those that change sporadically or with low rate, for example one every month. Dynamic data are those that change over time, with their specific data rate.

Due to the large difference among these two aspects they have to be managed in different manners. Moreover, a relevant role is covered in the city by private data coming from mobility and transport such as those created by Intelligent Transportation Systems, ITS, for bus management, and solutions for managing and controlling parking areas, car and bike sharing, car flow in general. Both open and private data may include real time data such as the traffic flow measure, railway and train status with respect to the arrival, parking areas status, Bluetooth tracking systems for monitoring people movements, and TV cameras streams for security and flow assessment.

Both PA sand mobility operators have large difficulties in elaborating and aggregating data to provide new services, even if they could have a strong relevance in improving the citizens' quality of life. Therefore, our cities, even those very active in smart city, are not so smart as they could be by exploiting a semantically interoperable knowledge base on the available data.

Static Data are largely created as open data from PA. They may be statistic information about the city, locations of point of interests (POIs), information about GOV services, etc. This information is typically accessible as public files in several formats, such as: SHP, KMZ, CVS, ZIP, XML, etc. For the points of interest, POI, the information recovered can be related to: museums, monuments, theaters, libraries, banks, express couriers, police, firefighters, restaurants, pubs, bars, pharmacies, airports, schools, universities, sports facilities, hospitals, emergency rooms, government offices, hotels and many other categories.

On the other hand, in most cases, the data sets are *not semantically interoperable*. The typical problems start with the management of the complexity of the several distinct input data sources, in terms of: formats, publishers and licenses (private or public data), method adopted to expose/publish the data by the data providers, etc. In this context, a relevant problem is the quality of data and their needs of reconciliation with the other information. This means to adopt a common strategy to make the data uniform and referring to the same entities. For example, a single street address (‘VIALE SPARTACO LAVAGNINI, 16’) can be referred in many different modalities depending on the rules used from the database from which they have been extracted (‘V.le Spartaco Lavagnini’, ‘v. S. Lavagnini’, etc.).

Therefore, the process of data ingestion for static data has to include activities (see **Figure** 2) of:

- **Ingestion of the data** set file: it may take files from web with HTTP/FTP protocols or make calls to web services authenticated or not. Each ingested data set has to be stored into a storage (implemented in this case with Hbase) to keep trace of the eventual changes over time of the static data, thus performing the versioning of the static data sets.
- **Quality Improvement**: it aims to increase the accuracy and consistency of the information ingested through a standard format identified for each specific field of the datasets. It makes and harmonization and reconciliation of the data. For example, different data providers use personal modalities or adopts different standards to describe a street, a Point of Interest, a kind of measure, a GPS coordinate, etc. In this case, the process of quality improvement, generate a new version of improved data directly into the Hbase storage. Please note that the quality improvement algorithm may radically change the data structure, for example splitting data in more fields, enriching record with more information (e.g., GPS location, CAP, etc.).
- **Triplification** (mapping): to produce a set of RDF triples compliant with the Km4City Smart City multi-ontology and saved in the RDF Storage (more details on the next section). The goal in fact is to map the data on a semantic model and store this information into an RDF store, to have data semantically most significant of the raw data ingested, passing through the ingestion and quality improvement phases described in section 3. The semantic Ontology used to produce the RDF data Knowledge and useful to make the sematic aggregation is Km4City [Km4City]. KM4City is a multi-ontology adopted in many European and National projects, such as Sii-Mobility, Replicate, Resolute [Sii-Mobility], [Replicate EU project, http://replicate-project.eu/], [Resolute EU project, www.resolute-eu.org].

Each data has its own ingestion process consisting of a number of scheduled ETL transformations (Extract, Transform, and Load) managed by the distributed scheduler, DISCES, see **Figure** 2. Typically, one for each phase. **DISCES**: Distributed SCE Scheduler consists of a set of distributed instances of running agents performing concurrent tasks on multiple servers (Virtual machines, Nodes). ETL transformations for static data are periodically executed to see if the data source has been changed, and in the positive case, the ingestion is updated.

On the other hand, most of the data sources descriptors may be affected by the above described problems and may also fit in model providing a super set of fields (such as: ID, description, category, location, street, etc.) which can be automatically ingested, and only some of them may be mandatory. For this reason, the DataGate tool has been created to allow users to provide data in terms of CSV files, automatically perform the activities of quality improvement and reconciliation. And since not all cases can be solve, the DataGate assists the user with a wizard to perform a limited number of activities for solving the remaining critical problems.

DataGate is an plugin module of CKAN [CKAN]. It provides support for automatically converting datasets of static POI into a Km4City compliant format and Knowledge Base. Thus, non-technical users can: i) upload dataset (churches, museums, hotels, hospitals, etc.) following a template in a csv format; ii) see data set improved automatically (e.g., extraction of latitude and longitude from street and civic numbers, augmenting the location with CAP, province, locality, and information taken from Web); iii) classify the information uploaded according to the Km4City ontology; iv) publish the dataset on the Datagate Portal in regularized format for promotion and further reuse; v) generate triples for the Knowledge Base. For example, the publishers can use the Service Map to see the points of interest in the map.

Dynamic Data are those that change over time, with their specific data rate and typically refer to static data. For example, a sensor is classified as static data when it is registered on a given position and data type; while it can be generated a lot of real time data without changing the initial registration of its ID and GPS location and data type (i.e., temperature). Most of the dynamic data are for example: weather forecast, value from environmental aspects, measures of pollution, traffic flow, parking status, people flow monitoring, events in the city about traffic and ordinances for city maintenance, IOT sensors, opening time and day for pharmacies, triage status, status of bridges, RTZ status, queue status of the GOV services, etc. They are strongly heterogeneous in terms of data structure and almost homogenous in referring to some static descriptor of the GPS point/entity at which they refer.

Also in this case, the ETL processes can be used and have to be scheduled as back office data ingestion process (see Figure 2). The scheduling by DISCES of these processes is much more computational consuming with respect to the processes for static data ingestion. In fact, real time ingestion can be performed with rates that are from few seconds to 1-2 times per week, for example. The results of the real-time data ingestion could impact on both:

- **Knowledge base**: when the data collected is not related to static information. For example, the real-time information about the bus line paths may change over time and typically refer to new paths (new position of bus stops, etc.) in the city and thus the Knowledge base has to be updated to allow city users discovering where the bus is passing.
- **Tabular storage**: when the data collected are instances of entities. For example, the value measured for a sensor at a given date and time. Belong to this type of data process also those arriving in push as: (i) stream that have to be collected with dedicated server processes and stored, such as data provided by Apps, data provided by Access Point monitoring people flow, etc., (ii) IOT data according to publish subscribe protocols, for example, LoraWAN, NGSI, SigFox, MQTT, AMQP, COAP, ETSI M2M, OneM2M, etc.

In first case, the final results are new triples in the KB, while in the second case, just new records in the storage. In both cases, the KB and the tabular storage have to be keep connected since the passage from geo located information to data and viceversa are mandatory.

In the above described cases, the adopted ETL tool has been Penthao Kettle.

3.2 The Storage

The data ingested have to be stored into a distributed storage with high performance and with fault tolerance capabilities to perform the data analytics, the access from dashboard and the access from mobile and web app via Smart City API. The data analytic processes, also produce additional knowledge and data that have in turn to be stored into a safe storage for contributing to the same services: dashboard, further analytics, business intelligence, Web and mobile Apps, etc. According to the above description, the variability, complexity, variety, and volume of these data flows make the data process of ingestion, aggregation, and data analytics as a "Big Data" problem. The variety and variability of data can be due to the presence of several different formats, and to scarce (or non-existing) interoperability among semantics of the single fields and of the several data sets.

The **storage architecture** depicted in Figure 2 comprehends: the RDF storage hosting the KB composed of the RDF triples and the Km4City ontology and a large distributed NoSQL database for all the real time tabular data described above.

The **RDF store** holds mainly static data, and dynamic data that change geolocation and geodescription overtime. A RDF stores can be: in-memory, native, non-memory non-native. In-memory RDF stores the RDF graph in main memory and hence could not be adopted for storing extremely large volumes of data. The native triple stores provide persistent storage with their own implementation of the databases. The non-native non-memory triples stores are set up to run on third party databases like MySQL, PostsgreSQL, Oracle. In the case of Firenze/Tuscany, the initial KB with only static data was of about 81 Million triples, with growth of 4 million triples per month. Actually, the number of triples in the RDF store is in the order of 300 million. An efficient RDF storage should offer both scalability in its data management performance and variety in its data storage, processing and representation. On the other hand, the KB has to be periodically cleaned since the historical data may create a reduction in performance [Bellini et al., 2016]. In our solution, the RDF Storage of the KB has been implemented by using a Virtuoso triple store. Virtuoso, is a native triple store available in both open source and commercial licenses. It provides command line loaders, a connection API, support for SPARQL and web server to perform SPARQL queries and uploading of data over HTTP.

The **tabular data** collecting real time data for referral information should be stored into a noSQL database that should be easily accessible for Data Analytics. This means to make them accessible for drill down via SQL, for example via Zeppelin or other database browser. On the other hand, most of the noSQL databases are not SQL compliant and thus languages such as R, Java, Python, ETL, etc., have to provide special plugin / extension to access them in read and write modalities. Also, having standard interface for query data permits the developer to avoid writing specific code for retrieving specific data. There are two dominants per type NoSQL databases: MongoDB (document oriented) and HBase (Wide Column Store). Wide column stores introduced by google Bigtable stores data in records with dynamic columns. Document stores, also called document-oriented database systems, are characterized by their schema-free organization of data i.e. records do not need to have a uniform structure (the types of the values of individual columns can be different for each record). MongoDB is an open-source document-oriented. It uses JSON, allowing for a schema less data model where the only requirement is that an id is always present. MongoDB's horizontal scalability is mainly provided using automatic sharding. Conversely MongoDB is doe not automatically treat operations as transactions and needs closed source solution to expose a standard interface SQL oriented.

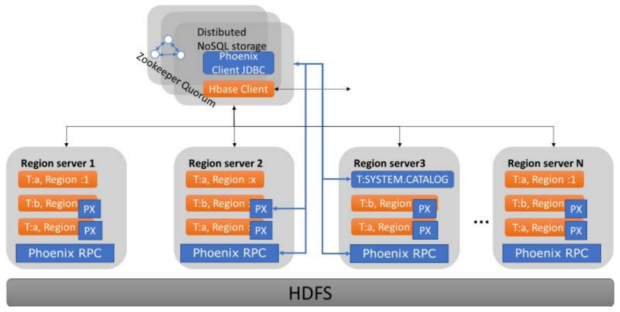

Figure 3. Sii-Mobility Detailed Architecture.

Apache HBase is a non-relational distributed database inspired on Google's BigTable. It has a fault-tolerant way of storing data and it is good for storing sparse data. Similarly to Google BigTable, it does not support full ACID semantics, although several properties are guaranteed.

In Sii-Mobility Architecture, Apache HBase has been adopted and deployed on an Apache Hadoop cluster which is configured in High Availability mode: 2 masters plus 4 slaves with automatic failover. High availability is obtained by distributing meta data filesystem information by distributed service i.e. journal nodes ensembles. The automatic failover is realized by failure detection and active node election mechanism implemented on a distributed configuration service i.e. Apache Zookeeper. Apache Zookeeper is a highly available service for maintaining small amounts of coordination data, notifying clients of changes in that data, and monitoring clients for failures. While for the ingestion phase a standard HBase deploy fulfill the above requirements, on the other hand it's inefficient when coping with dynamic data.

Moreover, querying data from HBase is implemented by scan operations which access all the table HBase does not have indexes as the MySQL database. The access pattern is based on row key. Orthogonal access patterns require a scan of the whole table (server-side), and then the application of additional filters (client-side). So, it is possible to apply custom filters, client-side, to the whole table to analyze a subset of the results but this does not reduce server-side IO costs. This mechanism inevitably involves high computational costs. To overwhelm this limit, the Distributed NoSQL Storage is implemented as Apache HBase with Apache Phoenix [Phoenix].

Apache Phoenix, that is an open source and relational database layer that is built on top of HBase. Apache Phoenix permits applications to store, retrieve or query millions of rows data from HBase by Structured Query Language (SQL) near real-time [Gupta et al., 2015], [Chrimes et al., 2015]. The advantage of Phoenix is that it achieves real time responses compiling the queries and statements from the client into a series of HBase 'intelligent' *scans*, *filters*, and *coprocessors* and then runs them to produce a selected result set. Phoenix allows to return therefore no longer the whole table but only the data requested by the client. Phoenix retrieve time is in the order of milliseconds for smaller queries and seconds when using millions of rows.

Phoenix provides JDBC (Java Data Base Connectivity) to JVM base clients and PQS for non-JVM base clients. PQS allows the non-JVM base clients to use a thin driver for the query plan, execution, and processing on an external server process to be scaled horizontally, independent of the client process as the query server is stateless. The thin driver is based on the Avatica framework, which provides an API between the client and the server [Avatica]. In Avatica, the server is an HTTP server and the client is a simple JDBC driver that allows the client to communicate over protocol buffers or JSON. Wire protocols provide the flexibility to have clients in non-JVM languages.

Phoenix is a relational database layer because it offers:

- *Transaction support*: has full ACID (Atomicity, Consistency, Isolation, Durability) semantics with the help of Apache Tephra for HBase row-level transactional semantics. Apache Tephra provides snapshot isolation of concurrent transactions by implementing multi-versioned concurrency control.
- *User-defined function*s: can be temporary or permanent user-defined or domain-specific scalar functions. Temporary functions are specific to a connection and are not accessible in other connections.
- *Secondary indexes and views*: are created for a large frequently accessed table when primary index sorting is not possible or hard to apply. Secondary indexes created on alternate row keys can allow point lookup, are much faster, and do not require a full scan on the table.

Phoenix supports view syntax as happens in standard SQL to enable multiple virtual tables to all share the same underlying physical HBase table.

To connect the ETL processes with the Distributed NoSQL Database, in the case of Dynamic datasets (*ETL Direct save* in Figure 2), it has been made an analysis related to the connections between Phoenix and Pentaho, from which emerges that: i) Pentaho uses an abstraction layer, called *shim*, that connects to the different Hadoop distributions; ii) *shim* is a small library that intercepts API calls and redirects or handles them, or changes the calling parameters. Because those shims are developed specifically for Hadoop distributions vendors such as Cloudera, Hortonworks and MapR, it was developed a plugin to query Phoenix cluster via JDBC with the version of HBase Hadoop actually installed. This plugin is implemented as a "user defined class" Pentaho object.

The plugin is a user defined class which essentially reads the value and the relative meta data, constructs the query and commits it on the NoSQL Database.

4. Experimental results

For the case of Florence/Tuscany area, we are addressing more than 800 different data sets sources of the 1550 available. Each open data ingestion process has to retrieve information and produce records in a noSQL DB for big data, logging all the information acquired to trace back and versioning the data ingestion. Then, the data have to be completed and improved in quality; and finally, data obtained are placed in tabular form, and triples RDF linked data form. At regional level, Tuscany Region provided a set of open data into the Mobility Integration Information Center of the Tuscany Region (MIIC), and provide integrated and detailed geographic information reporting each single street in Tuscany (about 137,745), and the locations of a large part of civic numbers, for a total of 1,432,223 (a wider integration could be performed integrating also Google maps and yellow/white pages). The solution has been adopted by the Florence municipality as the Smart City data aggregation tool. On which a number of additional projects have been based as: REPLICATE, RESOLUTE, etc.

The **static data** sets produced about 180.000 POIs each of which with complex set of data. In addition, a part of them (about 100) have been ingested by using DataGate that was not available since the beginning. Since its activation 85% of datasets for static data are ingested via the DataGate and thus the data ware house does not need to the programmed as new ETL process. Among those automatically ingested: cultural activities (libraries, churches, museums, theaters, monuments, etc.), hospitals, Wi-Fi, entertainments (beaches, …), accommodations (B&B, hotels, …), etc. in Florence, Venice, Bologna, Sardinia. More complex static data still have to pass via dedicated ETL processes such as: conversion from street graphs, limited traffic zones, cycling paths, etc. The static data are only a minimal part of the smart city complexity in terms of data ingestion.

For the Real Time data, every day about 5900 processes are executed. They include data from the ingestion of: bus tracking, mobility events, traffic flow sensors, parking, Apps positions and requests, Wi-Fi monitoring people flow, fuel station prices, weather forecast, environmental and pollution values, IOT sensors, first aid triage, public transportation lines updates, recharge station, time schedule for pharmacies and other services, etc., for about 1.8 Gbyte of new triples per day, 315.000 new data groups per day collected, integrated and linked to the graph database for Smart City API, data analytics, and Dashboards.

The typical success rate per day for the scheduled processes for data ingestion is close to the 94%. The faults are typically due to communication problems, the lack of communication with the server providing the data.

5. CONCLUSIONS

Smart city permits monitoring the city status connecting the different events that occur in a city at a given instant of time, provide suggestions to Public Administrations or other city authorities such as police, civil protection, hospitals, etc., permits to put in action city/region strategies and guidelines and obviously directly to the citizens. A smart city must cope with many kinds of data coming from different contexts which must be aggregated and integrated to have a general overview of the city. This means ensuring compatibility with multiple protocols from urban operators, handling open and private data, with the corresponding licenses, work with IOT/sensors/internet of everything, in cloud, perform predictions. Manage and aggregate all the aspects involved in a Smart City/Region/State means to solve a Big Data problem. This paper presented the work performed on defining Sii-Mobiility which is a smart city multi-tier architecture and identifying related issues and solutions. Moreover, the paper focuses on ingestion and aggregation phase motivating the adoption of polyglot persistence, addressing scalable solutions based on Hbase and Phoneix and at the same time collecting graphs and relationships into a graph DB with semantic acss and inference capabilities. This hybrid approach of multiple noSQL data store is a good compromise in having sophisticate queries formulated by using SPARQL and managing data into large scalable noDQL data store. To allow a better integration ETL processes have been endowed of tools for interfacing with Phoenix and thus with Hbase. Finally some figures have been discussed referring to Tuscany area in center of Italy.

6. ACKNOWLEDGMENTS

Our thanks to the MIUR, to the University of Florence and companies involved for co-founding of Sii-Mobility Project SCN 0112. Km4City is an open source technology of DISIT Lab.

7. REFERENCES

[Avatica] https://calcite.apache.org/avatica/

[Badii et al., 2016] C. Badii, P. Bellini, D. Cenni, G. Martelli, P. Nesi, M. Paolucci, Km4City Smart City API: an integrated support for mobility services, in: (SMARTCOMP) IEEE International Conference on Smart Computing, IEEE, 2016.

[Badii et al., 2017] Badii, C., Bellini, P., Cenni, D., Difino A., Nesi, P., Paolucci, M. Analysis and assessment of a knowledge based smart city architecture providing service APIs. Future Generation Computer Systems 75 (2017) 14–29. https://doi.org/10.1016/j.future.2017.05.001

[Badii et al., 2017] Badii, C., Bellini, P., Cenni, D., Difino, A., Paolucci, M., & Nesi, P. (2017, May). User Engagement Engine for Smart City Strategies. In Smart Computing (SMARTCOMP), 2017 IEEE International Conference on (pp. 1-7). IEEE.

[Barneers et al.] 5 Stars Open Data from Tim Barneers Lee. http://www.slideshare.net/TheODINC/tim-bernerslees-5star-open-data-scheme.

[Bellini et al., 2014b] P. Bellini, M. Benigni, R. Billero, P. Nesi, N. Rauch, Km4City ontology building vs. data harvesting and cleaning for smart-city services, Int. J. Visual Lang. Comput. (2014) http://dx.doi.org/10.1016/j.jvlc.2014.10.023.

[Bellini et al., 2014] P. Bellini, P. Nesi, A. Venturi, Linked Open Graph: browsing multiple SPARQL entry points to build your own LOD views, Int. J. Visual Lang. Comput. (2014), http://log.disit.org.

[Bellini et al., 2015] P. Bellini, I. Bruno, P. Nesi, N. Rauch, Graph Databases Methodology and Tool Supporting Index/Store Versioning, J. Visual Lang. Comput. (2015)

[Bellini et al., 2016] P. Bellini, L. Bertocci, F. Betti, P. Nesi, Rights enforcement and licensing understanding for RDF stores aggregating open and private data sets, in: second IEEE International Smart Cities Conference, ISC2 2016, Trento, Italy, SLIDES, 12 to 15 September 2016. http://events.unitn.it/en/isc2-2016.

[Bellini et al., 2017] Bellini, P., Cenni, D., Nesi, P., & Paoli, I. (2017). Wi-Fi based city users' behaviour analysis for smart city. Journal of Visual Languages & Computing, 42, 31-45.

[Bonino et al., 2015] Bonino, D., Delgado Alizo M.T., Alapetitey, A., Gilberty, T. ALMANAC: internet of things for smart cities. 3rd International Conference on Future Internet of Things and Cloud (FiCloud), Rome, Italy, August 2015. DOI: 10.1109/FiCloud.2015.32

[Castillo et al., 2014] P.A. Castillo, A. Fernández-Ares, P. García-Fernández, P. García-Sánchez, M.G. Arenas, A.M. Mora, G. Romero, V.M. Rivas, J.J. Asensio, J.J. Merelo, Studying individualized transit indicators using a new low-cost information system, in: Handbook of Research on Embedded System Design, IGI Global, 2014, pp. 388–407. http://dx.doi.org/10.4018/978-1-4666-6194-3.ch016, (Chapter: 16), January, Editors.

[Chrimes et al., 2015] Chrimes, D., Hu, W., Kuo, M., & Moa, B. (2015). Design and Construction of a Big Data Analytics Framework for Health Applications. SmartCity.

[CKAN] CKAN: http://ckan.org.

[Green Button Connect] Green Button Connect: http://www.greenbuttonconnect.com/.

[Gupta et al., 2015] Gupta, K., Sachdev, A., & Sureka, A. (2015). Pragamana: Performance Comparison and Programming Alpha-miner Algorithm in Relational Database Query Language and NoSQL Column-Oriented Using Apache Phoenix. C3S2E.

[IETF] IETF: https://www.ietf.org.

[KM4City] KM4City, http://www.km4city.org/

[Lin et al., 2013] F.J. Lin, Y. Ren, E. Cerritos, A feasibility study on developing IoT/M2M applications over ETSI M2M architecture, in: 2013 International Conference on Parallel and Distributed Systems, ICPADS, IEEE, 2013.

[Nesi et al., 2016] Nesi, P., Badii, C., Bellini, P., Cenni, D., Martelli, G., & Paolucc, M. (2016, May). Km4City Smart City API: an integrated support for mobility services. In Smart Computing (SMARTCOMP), 2016 IEEE International Conference on (pp. 1-8). IEEE.

[OpenDataSoft] OpenDataSoft: https://www.opendatasoft.com/.

[Patti et al., 2016] Patti, E. and Acquaviva, A. IoT platform for Smart Cities: requirements and implementation case studies. 2nd IEEE International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI 2016), Bologna, Italy, 7-9 September 2016. pp. 1-6 IEEE DOI:10.1109/RTSI.2016.7740618

[SIIMOB-DE1-1] http://www.sii-mobility.org/images/deliverables/Sii-Mobility-DE1-1aanalisi-requisiti-casi-uso-v3-0-pub-final.pdf

[Sii-Mobility] Sii-Mobility EU project, www.sii-mobility.org

[Sinaeepourfard et al., 2016] Sinaeepourfard, A., Almiñana, J. G., Masip, A. and Marin-Tordera, E. Estimating Smart City Sensors Data Generation Current and Future Data in the City of Barcelona. 2016 International *Conference* on Information and Communication, IEEE. DOI: 10.1109/MedHocNet.2016.7528424

[SPARQL] SPARQL: https://www.w3.org/TR/rdf-sparql-query/.

[Swetina et al., 2014] J. Swetina, et al., Toward a standardized common M2M service layer platform: Introduction to oneM2M, IEEE Wirel. Commun. 21 (3) (2014) 20–26.

[Virtuoso] Virtuoso, https://virtuoso.openlinksw.com