Data Ingestion and Inspection for Smart City Applications

Pierfrancesco Bellini, Daniele Bologna
Department of Information Engineering, DISIT
University of Florence, Italy
pierfrancesco.bellini, Daniele.bologna@unifi.it

Qi Han
Department of Computer Science
Colorado School of Mines, Golden, CO 80401 USA
qhan@mines.edu

Paolo Nesi, Gianni Pantaleo
Department of Information Engineering, DISIT
University of Florence, Italy
paolo.nesi, gianni.pantaleo@unifi.it

Michela Paolucci
Department of Information Engineering, DISIT
University of Florence, Italy
michela.paolucci@unifi.it

***Abstract*—Smart cities are distributed heterogeneous systems of systems connected to each other via a variety of heterogeneous data streams involving multiple stakeholders and organizations. This complexity is reflected also in the data that have to be managed to provide a concrete and useful real time service to the citizens. The data ingestion phase is critical for the whole services, since it has to preserve the information, connect the new data with old data and establish right connections with city entities. This paper describes data ingestion and inspection in the Snap4City open source scalable Smart aNalytic APplication builder, with a specific focus on how heterogeneous data is represented, how its quality is inspected, and how to develop ingestion procedures in an efficient manner. The Snap4City ingestion processes are based on a semantic and unified data ingestion model, capable of aggregating different types of data. A performance comparison of different data ingestion modalities is presented.**

***Index Terms*—smart city, data ingestion, data inspection**

I. Introduction

Smart cities are distributed heterogeneous systems of systems, ranging from IoT Networks to front end distribution. Smart Cities are managed by multiple stakeholders and organizations sometimes with competing objectives, and are exploited by city users including citizens, tourists, commuters, students, operators, etc. One major challenge for smart cities solutions is to collect and manage heterogeneous multi-dimensional data sources, ensuring interoperability of data represented in any format and transmission protocol. Many applications in the smart city context [4], [7], [10] have been developed, but they often take a vertical approach and focus on a specific domain such as Smart Power Grids [2], smart parking [3], smart meters [10], [12], or roadside assistance [5]. A typical smart city system presents a multitude of data providers, data exchange modalities and licenses. These include IoT/IoE, Open Data portals, social media, private and/or public data, GIS, city utilities, etc. To manage all these aspects, many IoT systems and big data frameworks put emphasis on the life cycle of data [3], [6]. Give importance to the life cycle approach is fundamental to identify problems in data gathering, thus having the possibility to contact a data provider in case of missing or low quality data, or tracing back to the data source to assign the right user license on a single data, etc. [2], [3]. In smart cities, a big data infrastructure must be instrumented with advanced data gathering flow processes taking information from different data providers and considering a set of big data requirements, such as those classified in [9], [11], [13]: i) volume, ii) velocity, iii) variety, iv) variability and (v) veracity (data quality). Additional aspects are related to semantic data aggregation, data redundancy, fault tolerance, licences of use, data protection, real time queries on the Knowledge Base, data analytic and visualization or GIS-based visualization, drill down on data, etc. The work presented in this paper focuses on data ingestion and inspection developed in the Snap4City framework, [14], [15]. While most existing works have adopted a simple NoSQL for data management, in Snap4City a multimodal approach is developed, performing data ingestion and applying semantic reconciliation strategies to uniformly take both dynamic static data from traditional services such as rest API, WS, FTP via PULL protocols as well as IOT data using PUSH protocols. This paper describes the process of ingestion of the heterogeneous data managed by the Snap4City platform. Moreover, a unified data ingestion model to accelerate these processes is described. In Section II, an overview of the Snap4City system is introduced; in Section III, the data Ingestion process model is described. Section IV provides a detailed description of the data ingestion model. In Section V, a performance comparison of different data ingestion modalities is provided. Section VI concludes the paper.

II. Snap4city System Overview

Snap4City has been developed to provide many online services and suggesting guidelines to involve all different kinds of

The authors would like to thank the European Union's Horizon 2020 research and innovation program for funding the "Select4Cities" PCP project (within which the Snap4City framework has been supported) under grant agreement No 688196, and also all the companies and partners involved. Snap4City and Km4City are open technologies and research of DISIT Lab https://www.snap4city.org

978-1-7281-6997-2/20/$31.00 ©2020 IEEE
DOI 10.1109/SMARTCOMP50058.2020.00052

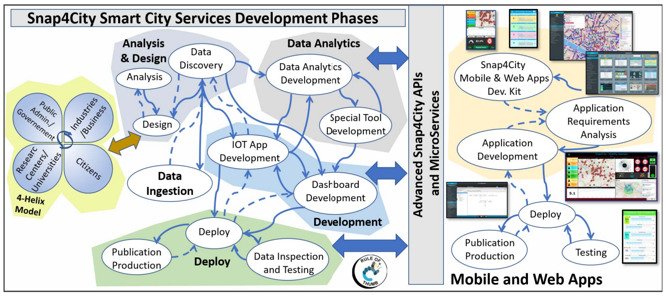


Fig. 1. Snap4City Smart City Services Development Phases.

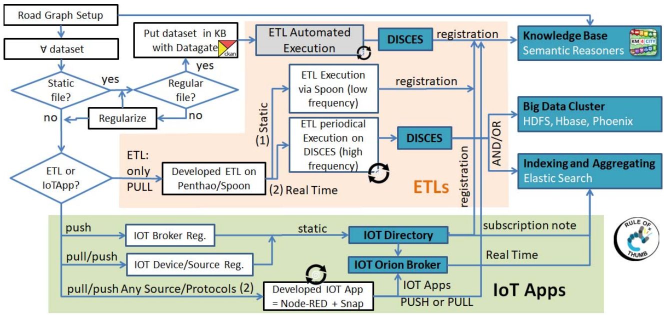


Fig. 2. Snap4City Data Ingestion Diagram Flow.

organizations (e.g., Research Centers and Universities, small and large industries, public administrations and local governments) and citizens (city operators, companies, tech providers, corporations, advertisers, city users, community builders, etc.) [14], [16]–[19]. It adopts a Quadruple Helix approach [16], [17] as shown in Fig.1. The Snap4City process consists in the analysis of the requirements and needs involving all the city decisors and data providers to establish what types of data are needed and/or available. After that the data ingestion processes are realized and described in details in section III. Semantic integration and standardization is then carried out and is at the basis of the Data Analytics and Algorithm Development processes, such as predictions and high performance analysis. Snap4City is compliant with GDPR (General Data Protection Regulation of the European Commission), it is capable to manage a variety of data having different licensens. Snap4City has been applied in many Italian (Firenze, Pisa, Livorno, Prato, Lonato, etc.) and European cities (Antwerp, Helsinki, Santiago De Compostela) and in their surrounding geographical area (e.g. Tuscany, Sardinia, Lombardia but also Belgium and Finland) [15].

III. Data Ingestion Process Model

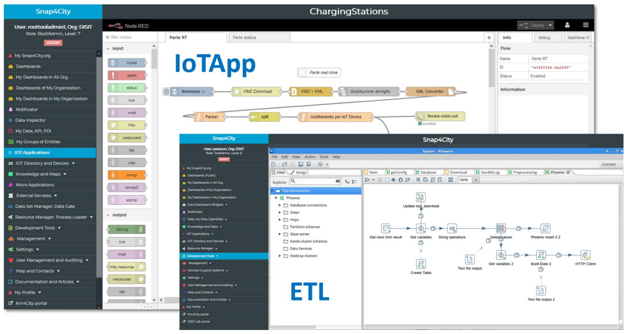
Fig.2 shows the Snap4City Data Ingestion Diagram Flow [14], [15]. In a Smart City, the first action of the Snap4City data ingestion process is the Road Graph Setup collecting data on streets coming from city government as well as open datasets (e.g. Open Street Map). In this way Points of Interest (POI), Sensors, Citizens, etc., can be connected to road graphs and located in a specific place of the city, not only based on their coordinates but also on the streets and civic numbers. After this, it is necessary to understand if a dataset is i) only static, or ii) has also some dynamic fields that can change in future. The first case is the easiest one, a typical sample are the Point of Interests (POIs), data that are usually available in the Official Open Data Portals in Europe, most of them based on CKAN. For this reason, a solution to automate the process from CKAN to KB via a customized plugin (DataGate) has been integrated in Snap4City. It regularizes the open data via a template to be filled by the data providers, after that the data are processed by an Extract Transform and Load process (ETL), capable to map each data in the Km4City multi-ontology. The ETL is executed on a distributed Scheduler called DISCES and the data are automatically put in the Snap4City Knowledge Base [14], [15]. The second case is more interesting, as a non-regular dataset can have both static and dynamic info. Many kind of data belong to this case: a car park monitoring system has both a fixed location and registers data every minute (i.e. number of free slots, the amount of time each slot is free or occupied); a sensor for registering the number of people coming in a museum has a fixed location and counts the number of people every second; an air quality sensor placed on a bus continuously moves as the bus moves and takes measurements in real time, etc. If the data are not regular, different methodologies to classify/manage/exchange data (e.g Push or Pull) must be adopted, via ETL or IoTApp. In this paper, a relevant part is devoted to the comparison of these different ingestion tools, making an evaluation in terms of cost, complexity, and development time as shown in Fig.2.

A. ETL Data Ingestion

For the development of ETL processes, the Pentaho Kettle Open Source tool has been integrated in the system. For each dataset, two ETLs processes are created: (1) Static, addressing fixed aspects and ingesting them into the KB; and (2) Periodic, that is put in execution on the DISCES, depending on the data frequency variability. The data storage can be implemented via: a) a Big Data Cluster (based on HDFS, HBase, Phoenix); or b) an Indexing and Aggregating tool (e.g., based on Elastic Search). Each solution has its pros and cons, but in both cases replica and federations are set up, with vertical and horizontal scaling, thus creating a large data store with some indices. In both cases, queries are performed by using NoSQL approaches via API.

B. IoT Data Ingestion

IoT data is typically sent in push, using a publisher/subscriber protocol. IoT devices are registered in an IoT Broker which is registered on the Snap4City IoTDirectory. When a new IoT Device is created: i) a set of static data is registered on the KB; then ii) a command to the storage system for the subscription to the corresponding IoT Broker is sent. In case of HBase, for each new entity, a specific process is set up for writing into the storage, and it is implemented by



Name Parte RT
ID "e0ff088b.fa2268"
Status Enabled

Fig. 3. Ingestion via NodeRED IotApplications and Pentaho ETL processes in the Snap4City Portal.

using ETL or IoTApp. On the contrary, for Elastic Search, a scalable NIFI Apache ingestion process is implemented to automatically subscribe the IoT Brokers on all its devices, and feed the Elastic Search engine, thus creating the data shadow for IoT data. In our system, we have adopted an IoT App flow for registering the data model, as an IoT Device, on both the IoT Broker and IoT Directory via the Snap4City APIs. A second IoT App flow registers all the metadata and descriptors for modeling the new entry into the KB. When a new dataset needs to be ingested, if the methodology adopted is ETL or IoTApp, an ad-hoc semantic mapping to connect each sensor/IoTDevice, POI, etc. to the KM4City ontology is realized. The mapping generates a set of RDF Triples based on the KM4City classes and properties and then adds them to the Snap4City KB, realized in Virtuoso. The IoT Applications can be used for data ingestion both in pull and push, while ETL collect data only in PULL. In case of ETL, the work of creating the triples must be done by the developers (using Karma data integration tool). If the ingestion method adopted is IoTApp, the most relevant triples are automatically created by the system and added to the KB when each sensor is registered (the registration is done with an easy to use web tool also in bulk), then developers are free to add other specific RDF triples via IoTApp. Developers are supported all the time, thanks to the Living Lab and co-creation activities available on the Snap4City platform.

C. Formal Model: IoT Ingestion vs Ontology

The following describes the semantic connections that are automatically added when a user registers his/her new sensor or IoT Device.

```
km4cr:eCharging_18XP22
  geo:long "11.261270001652804"^^xsd:float;
  geo:lat "43.77000404722702"^^xsd:float;
  http://schema.org/name "eCharging_18XP22";
  rdf:type sosa:Sensor;
  rdf:type km4c:Charging_stations;
  ssn:implements km4cr:iot/ChargingStation;
  km4c:hasAttribute
   km4cr:iot/orionFirenze-UNIFI/Firenze/eCharging_18XP22/chargingState;
  km4c:hasAttribute
   km4cr:iot/orionFirenze-UNIFI/Firenze/eCharging_18XP22/chargingStateValue;
  km4c:hasAttribute
   km4cr:iot/orionFirenze-UNIFI/Firenze/eCharging_18XP22/dateObserved;
  km4c:hasAttribute
   km4cr:iot/orionFirenze-UNIFI/Firenze/eCharging_18XP22/stationState;
  km4c:hasAttribute
   km4cr:iot/orionFirenze-UNIFI/Firenze/eCharging_18XP22/stationStateValue;
  sosa:observes km4cr:value_type/charging_station_state;
  sosa:observes km4cr:value_type/charging_state;
  sosa:observes km4cr:value_type/timestamp;
  ssn:hasSystemCapability
   km4cr:iot/orionFirenze-UNIFI/Firenze/eCharging_18XP22/systemCapability;
  iot-lite:exposedBy km4cr:iot/orionFirenze-UNIFI;
  km4c:protocol "ngsi";
  km4c:format "json";
  km4c:model "ChargingStationModel";
  km4c:producer "Comune di Firenze";
  km4c:macaddress "...";
  km4c:organization "Firenze".

#for each attribute
km4cr:iot/orionFirenze-UNIFI/Firenze/eCharging_18XP22/chargingState
  rdf:type km4c:DeviceAttribute;
  km4c:order 1;
  km4c:data_type "string";
  km4c:value_type km4cr:value_type/charging_state;
  km4c:value_name "chargingState";
  km4c:value_unit "-";
  km4c:value_refresh_rate "900";
  km4c:different_values "0";
  km4c:value_bounds "unspecified";
  km4c:editable "false"^^xsd:boolean;
  km4c:disabled "false"^^xsd:boolean.

#broker description
km4cr:iot/orionFirenze-UNIFI
  rdf:type km4c:NGSIBroker;
  iot-lite:endpoint "http://orion:1026";
  km4c:created "2019-10-28 10:01:53";
  schema:name "orionFirenze-UNIFI".
```

Additional triples can be manually added, for example related to the Graph street, as the following:

```
km4cr:iot/orionFirenze-UNIFI/Firenze/eCharging_18XP22
    schema:streetAddress "VIA GIUSEPPE VERDI" .
km4cr:iot/orionFirenze-UNIFI/Firenze/eCharging_18XP22
km4c:houseNumber "15" .
km4cr:iot/orionFirenze-UNIFI/Firenze/eCharging_18XP22
    schema:addressRegion "FI" .
km4cr:iot/orionFirenze-UNIFI/Firenze/eCharging_18XP22
   schema:addressLocality "FIRENZE" .
km4cr:iot/orionFirenze-UNIFI/Firenze/eCharging_18XP22
   km4c:isInRoad km4cr:RT04801703173TO .
```

IV. Unified Data Model

Data inspection for smart cities applications is faced with multiple challenges. Data sources are heterogeneous and diverse, data may come from stationary or mobile sensors, social media, web pages, etc. This large amount of data can be stored or managed in streaming. Different data storage models may be preferred with respect to data lakes, also basing on the data arrival and data update frequencies. Moreover, a wide variety of data consumers is present: some may use the data for offline processing, others conduct online processing; data analytic tasks are also different, some for visualization and others for data mining and still others for prediction and early warning. Considering all these challenges, in the smart city back office management it is important to have a unique model and tool, accessing and harmonizing all the information. To do this it is necessary to map and connect data providers, technical description, processes used for ingestion, ingestion status, quality level, relationships with other cities' elements, processes and tools used to collect data, data analytic, dashboards on which the data are shown. For example, a single sensor value may be received in push mode, the sensor is related to an IoT Device that may also have other sensors attached and it is managed by one or more IoT Brokers. The sensor and the IoT Device, can be shown on Dashboards or used by some data transformation or data analytic process. If something happens to the process, the manager should be able to guide the technicians to solve the problem quickly, suggesting to check the data provider, data ingestion, communication channels, or database, etc. Data can enter the system in different modalities, so data inspector of Snap4City manages the data according to classification categories of the data, High Level Type (HLT). The classification of HLT is relevant, since the model unification aims to strongly reduce the time for data discovery and composition when dashboards are created. The HLT or kind of **Sensors** can be conceptually classified into the following classes, while their usage is transparent for users:

- Sensor-ETL data collected "from ETL" processes running in a periodical modality thanks to the presence of the DISCES schedule. These are typically data collected via PULL by server;
- Sensor-IoT sending data "From IoT Devices to the knowledge base" via PUSH. These are data ingested by IOT Brokers with a publish-subscription protocol.
- Sensor-IoT sending data "From Snap4City Dashboards to IoT Applications."
- Sensor-Actuator which are data messages passing "From Snap4City Dashboards to IoT Applications." IoT Applications or Brokers are capable of sending actuation produced by the user via graphic widgets: switch, keypad, button, etc.

The major HLTs are described in the following in order of relevance: sensors, MyKPI, POI and MyPOI, Heatmaps, Events, External Services, Special, WFS (Web Feature Service) and WMS (Web Map Service). The HLT called **MyKPI** are geo-localized Key Personal data over time and have a time stamp and a new GPS coordinate. MyKPI provides some metadata to allow the unified model. MyKPIs are typically used for modeling sensors that can be moved in the city, e.g., a pax-counter used to count the number of people in a specific area of the city such as in a building, a museum, a district; a PM10 located on top of the buses. MyKPIs are used on dashboards/widgets for tracking (e.g., mobile sensors, on-board units of cars and buses, mobile apps) and showing the sensors position in a map to better understand the status in a period of time and develop new city strategies. The **POI** and **MyPOI**, are Point Of Interests and are typically used to model static data (a fixed GPS positions). For example, the position of restaurants, museum, benches, etc. are static information typically modeled as POI or MyPOI. POIs are typically collected from Open Data of the city and region. The **Heatmaps** are matrices graphically representing the geo-distribution of specific values. In most cases, the heatmaps are expected to be depicted according to a regular grid, but sensors are located in compliance with the city street graph, so data sources are from non-regular grid of sensors. The regular view is produced by some algorithms.

The **Events** (or Complex Events) can be generated by operators, tickets, policeman, and planned workers in the city. They are typically classified and thus can be shown on Dashboard according to different filters and status.

The **External Services** are typically links to third party's web pages and services, which can be useful to directly access to specific applications and services of the city such as: traffic light management, waste management, general administration of the ticketing system.

The **Special** are complex data records describing a situation with an had-hoc semantic. They are typically represented in dashboards with a dedicated widgets providing hight interactivity levels. Typical samples could be: the civil protection alert coding messages, the status of a parking area with an animated representation of the parking lots, the status of a gate, etc.

The **WFS and WMS** are data which may consist of direct links to the end point of a GIS WFS service providing a JSON including GIS data obtained on the fly. The connection allows access to the GIS data and also provides data to the GIS server. For example, a connection may be with a GeoServer, or an ArcGIS Server. This kind of data may include POI, shapes, images, orthomaps, and a range of structured information.

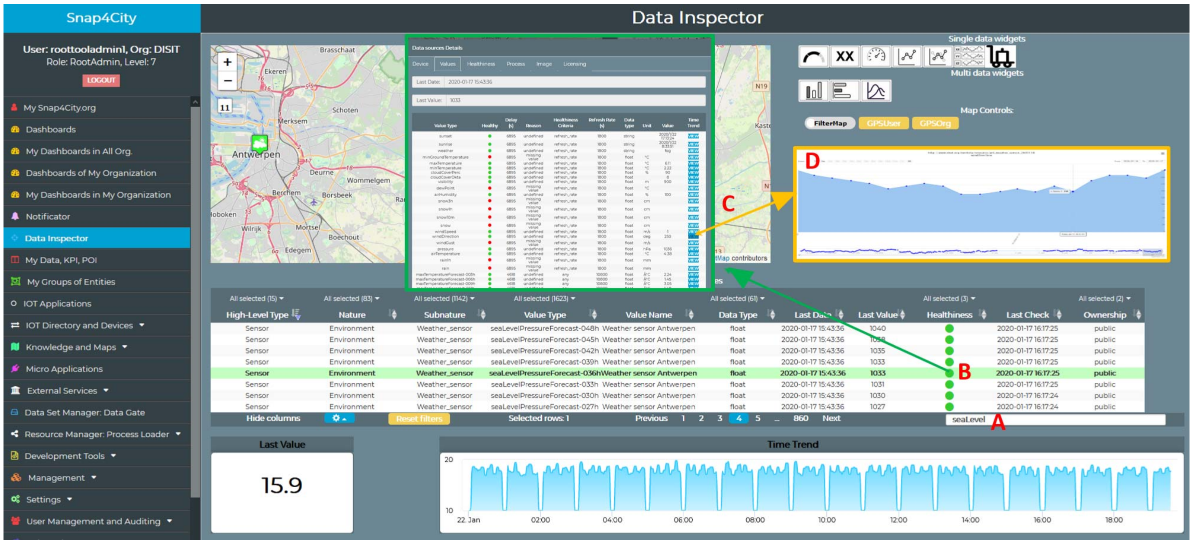
Fig.4 shows the data inspector tool that models device descriptor, values and links with other city entities, values over time, and value of the other sensors of the same father device, process involved for its ingestion, eventual image and licensing details.

A. HLT Metadata

The unified data model describes HLTs by using the following metadata for indexing them in faceted index. in addition, the faceted index is used in Data Inspector and Dashboard Wizard to facilitate the single data identification during data inspection and dashboard creation. In order to enable the faceted search, a classification is performed classifying the metadata into: semantic, technical, healthiness, ownership and licensing. In addition, other details are needed to manage data sources such as ingestion processes, data provider, historical values, eventual images and links to the tools for managing them in case of errors or problems in the data ingestion. The **Semantic** aspects include Nature and SubNature to classify the data and take trace of the relation with the Km4City multi-ontology classes such as mobility, energy, government, environment, etc. In fact, Nature and SubNature are taken from the KM4city ontology.

The **Technical** aspects include GPS coordinates, Value Name (the name of data), Value Type (e.g. temperature, velocity), Value Unit depending on the value type selected (e.g., Km/h or m/s for the velocity), data value (the actual value of a variable), Data Type (data format such as integer, binary, boolean, date, date/time, float, html, url, vector, webpage, wkt, xml, etc.). Value Type, Value Units and Data Types are taken from a well-defined dictionary of coherent terms. In the technical aspects, are also recovered: the last valid value, the date and time when data was obtained, a possible view of the data belonging to the same time series, i.e., the historical data of the same device or source. In Fig.4A, a traffic flow sensor (e.g. 'METRO640') can be selected using textual search or setting the filters on the metadata, then a set of possible HLT or metrics related to it are accessible. To access the value for a given sensor, a click on the green circle and the 'Data Source Detail' is popped up (Fig.4.B) is needed. Moreover, a click on one of the metrics (Fig.4.C) in the 'concentration', allows to access to the Data History Manager is shown.

Another relevant aspect is related to the data **Healthiness** due to the large variability of the HLTs. To assess the data healthiness a set of metadata are defined. The healthiness



High-Level Type	Nature	Subnature	Value Type	Value Name	Data Type	Last Date	Last Value	Healthiness	Last Check	Ownership
All selected (15)	All selected (83)	All selected (1142)	All selected (1623)		All selected (61)			All selected (3)		All selected (2)
Sensor	Environment	Weather_sensor	seaLevelPressureForecast-048h	Weather sensor Antwerpen	float	2020-01-17 15:43:36	1040	●	2020-01-17 16:17:25	public
Sensor	Environment	Weather_sensor	seaLevelPressureForecast-045h	Weather sensor Antwerpen	float	2020-01-17 15:43:36	1038	●	2020-01-17 16:17:25	public
Sensor	Environment	Weather_sensor	seaLevelPressureForecast-042h	Weather sensor Antwerpen	float	2020-01-17 15:43:36	1035	●	2020-01-17 16:17:25	public
Sensor	Environment	Weather_sensor	seaLevelPressureForecast-039h	Weather sensor Antwerpen	float	2020-01-17 15:43:36	1033	●	2020-01-17 16:17:25	public
Sensor	Environment	Weather_sensor	seaLevelPressureForecast-036h	Weather sensor Antwerpen	float	2020-01-17 15:43:36	1033	●	2020-01-17 16:17:25	public
Sensor	Environment	Weather_sensor	seaLevelPressureForecast-033h	Weather sensor Antwerpen	float	2020-01-17 15:43:36	1031	●	2020-01-17 16:17:25	public
Sensor	Environment	Weather_sensor	seaLevelPressureForecast-030h	Weather sensor Antwerpen	float	2020-01-17 15:43:36	1030	●	2020-01-17 16:17:24	public
Sensor	Environment	Weather_sensor	seaLevelPressureForecast-027h	Weather sensor Antwerpen	float	2020-01-17 15:43:36	1027	●	2020-01-17 16:17:24	public

Hide columns · Reset filters · Selected rows: 1 · Previous 1 2 3 4 5 … 860 Next · seaLevel

Fig. 4. Data Inspector in Snap4city.

criteria, depends on the type of data and must be specified by the responsible who ingests the data in the system. In some cases more than one method may co-exist and can be:

- **refresh rate** (in seconds): frequency at which a new value is expected to have. For example, if the frequency is 900 seconds and if the last value has been updated more than 900 seconds ago, data is considered old and a problem is reported. The refresh rate can also be regarded as the maximum duration between a valid value/sample and the next.
- **different values** (expressed as an integer N). This is a field useful to check the variability of the data values. If a value is the same for more than N times/samples, a problem is reported.
- **value bounds** are possible max and min value bounds. If a value is over the max or under the min, a problem is reported.

According to the model above, in order to estimate the healthiness, it is mandatory to have a precise timestamp of the last value obtained as collected in the set of technical aspects of the data. In the **Ingestion Process details**, it is necessary to associate various information with each single data item that may allow the back office personnel to understand how the data is ingested and processed. In fact, once an healthiness problem is detected, the administrator should be directly engaged in solving the problem inspecting the data ingestion processes and workflows. To this end, lots of information needs to be associated with each data item and in particular:

- **Data Source**, describes how the data are ingested into the platform, which may include Datagate, IoTApp, ETL, Java, IoT Device, Dashboards, etc.
- **Links to direct access the services** for managing data and/or external web pages. Each data may have one or more links such as Service Map, IoT Broker, DISCES scheduler, External Open Web Pages or Snap4City Dashboards. Following the links a data preview is visible.
- **Processes**, used to ingest the data and/or to manipulate them for data analytic. Differs on the Data Source type managed. Moreover, the data series can be stored into HBase or Elastic search noSQL data stores or both. In both cases, different procedures and tools are used to verify if the data have been correctly ingested or not.

The last aspect is related to the **Ownership and Licensing**. It is mandatory for any data city administrator to have a clear view of the ownership of the data and how these data can be used. In case of disputations and uncertainties it is fundamental to have a direct link to the data responsible holding the data ownership. In the Snap4City model, an Organization is typically related to a geo-political area, for example: Firenze and Tuscany, Helsinki and Finland, Antwerp and Belgium, Sardegna, etc. According to GDPR, all the data ingested start as strictly private and the eventual access to them is granted by instantiating specific delegations. Basing on the license, it is possible to delegate the access to the data that can be: private, public, accessible only for the creator or for a selected user or group of users.

V. Performance Evaluation

The Snap4City Data Ingestion process is implemented to ingest data using three main different procedures: Datagate, ETL processes, and IoTApps. Table I shows a comparative analysis of these three acquisition methods. Datagate is an ingestion tool for non-technical users. The metric considered for the comparison is the number of datasets uploaded. As of now 15,968 datasets are present in DataGate, of which only 3.8% have been uploaded following the Snap4City standard template with the semantic mapping on Snap4city. On the contrary, both the ETL processes and the IoTApps are managed by Snap4city registered users who are more technical. Most of the datasets

TABLE I
COMPARATIVE ANALYSIS AMONG DATAGATE, ETLS, IOTAPPS.

	Datagate	ETL	IoTApp
types of data managed	S	S, P	S, P, RT
Data protocol types managed	PULL	PULL	PULL&PUSH
Scheduling	external	external	internal
Flows to manage N instances of the same dataset	N	N	1
Users' technical level	without	medium/high	low
Development time	1,2 hours	1, 2 weeks	3, 4 days
Semantic (KM4City)	standard template	ad hoc (manual)	ad hoc (semi-automatic)
Developed number	1334 datasets	162	76
Mean number of blocks	0	120.333	27,67
Mean number of lines of code	0	275	229

managed in this case provide both static and dynamic data. The dynamic data can be periodical (P) or real time (RT). A distinction on the required technical level of the programmers must be done: for the development of ETL, the level has to be medium/high, while for the IoTApp, it is designed for people with a very low technical level. In fact, the time spent to ingest a dataset developing and scheduling an ETL process is about one or two weeks. For the same dataset with the IoTApp, the development time is three or four days. Both ETL processes and IoTApp are based on block or visual programming. For this reason a comparison among the two methodologies was performed on the number of blocks used and the lines of code written. It is also relevant that in the case of ETL processes (and consequently also for Datagate that is based on many instances of an ETL), an external scheduler is used, while for IoTApps an internal scheduler can be set. The presence of the internal schedule in the IoTApps also decreases the complexity when a set of different instances for the same IoTApp (or ETL) is managed. In the management of IoTApps, the scaling is provided using an elastic management of containers based on Mesos/Marathon frameworks.

VI. CONCLUSION

In this paper, a unified data ingestion model developed for Snap4City is described. A comparison among the different data ingestion methodologies adopted in Snap4City is conducted in terms of data structures, transmission protocol used, static or dynamic information, etc. The comparison shows that the Datagate tool is easy to use even in the presence of massive upload of i) static data that mainly need to be stored and not to be reworked to provide new knowledge; and 2) static data that have a standard set of metadata. In case of dynamic dataset, the best methodology is IoT Apps considering time consumption, implementation complexity, and semantic mapping.

REFERENCES

[1] A. A. Ghaemi, “A Cyber-Physical System Approach to Smart City Development”, 2017 IEEE International Conference on Smart Grid and Smart Cities, Singapore, July 2017.

[2] Q. Zhoua, Y. Simmhanb, V. Prasanna, “Knowledge-infused and consistent Complex Event Processing over real-time and persistent streams”, Future Generation Computer Systems 76 (2017) 391–406.

[3] A. Hefnawy, A. Bouras, C. Cherifi, “IoT for Smart City Services: Lifecycle Approach”, 2nd IEEE International Conference on Cloud Computing and Internet of Things(CCIOT2016), Cambridge, United-Kingdom., pp.55, March 2016, 10.1145/2896387.2896440.hal-01531630

[4] M. Ge, H. Bangui, B. Buhnova, “Big Data for Internet of Things: A Survey”, Future Generation Computer Systems, May 2018, DOI: 10.1016/j.future.2018.04.053

[5] S. K. Datta, C. Bonnet, “Next-Generation, Data Centric and End-to-End IoT Architecture Based on Microservices”, 2018 IEEE International Conference on Consumer Electronics, Asia (ICCE-Asia), June 2018. DOI: 10.1109/ICCE-ASIA.2018.8552135

[6] L. F. Rahmana, T. Ozcelebia, J. Lukkiena, “Understanding IoT Systems: A Life Cycle Approach”, Procedia Computer Science, Volume 130, 2018, pp. 1057-106, DOI: https://doi.org/10.1016/j.procs.2018.04.148

[7] Z. Lv, X. Li, H. Lv, W. Xiu, “BIM Big Data Storage in WebVRGIS”, Published in: IEEE Transactions on Industrial Informatics (Early Access), 13 May 2019, DOI: 10.1109/TII.2019.2916689

[8] A. A. Munshia, Y. A.-R. I. Mohamed, Big data framework for analytics in smart grids, Electric Power Systems Research, 151 (2017) 369–380, DOI: http://dx.doi.org/10.1016/j.epsr.2017.06.006

[9] S. Nadal, V. Herrero, O. Romero, A. Abelló, X. Franch, S. Vansummeren, D. Valerio, “A software reference architecture for semantic-aware Big Data systems”, Information and Software Technology 90, pp. 75–92, 2017, DOI: https://doi.org/10.1016/j.infsof.2017.06.001

[10] I. A. T. Hashema, V. Changb, N. B. Anuara,K. Adewolea, I. Yaqooba, A. Gania, E. Ahmeda, H. Chiroma, “The role of big data in smart city”, International Journal of Information Management, Volume 36, Issue 5, October 2016, pp. 748-758, DOI: https://doi.org/10.1016/j.ijinfomgt.2016.05.002

[11] W. Inoubli, S. Aridhi, H. Mezni, M. Maddouri, E. M. Nguifo, “An experimental survey on big data frameworks”, Future Generation Computer Systems, Volume 86, September 2018, Pages 546-564. DOI: https://doi.org/10.1016/j.future.2018.04.032

[12] T. Wilcoxa, N. Jinb, P. Flachc, J. Thumimd, “A Big Data platform for smart meter data analytics”, Computers in Industry, Volume 105, February 2019, pp. 250-259, DOI: https://doi.org/10.1016/j.compind.2018.12.010

[13] M. Babar, F. Arif, M. A. Jan, Z. Tan, F. Khan, “Urban data management system: Towards Big Data analytics for Internet of Things based smart urban environment using customized Hadoop”. Future Generation Computer Systems, Volume 96, July 2019, Pages 398-409DOI: https://doi.org/10.1016/j.future.2019.02.035

[14] C. Badii, E. G. Belay, P. Bellini, D. Cenni, M. Marazzini, M. Mesiti, P. Nesi, G. Pantaleo, M. Paolucci, S. Valtolina, M. Soderi, I. Zaza,“Snap4City: A Scalable IOT/IOE Platform for Developing Smart City Applications”, IEEE Smart City Innovation, China 2018. DOI: https://ieeexplore.ieee.org/document/8560331/

[15] P. Bellini, P. Nesi, M. Paolucci, I. Zaza, “Smart city architecture for data ingestion and analytics: Processes and solutions”, IEEE 4th International Conference on Big Data Computing Service and Applications, BigDataService 2018, pp. 137-144, March 2018. DOI: 10.1109/BigDataService.2018.00028

[16] P. Nesi, M. Paolucci, “Supporting Living Lab with Life Cycle and Tools for Smart City Environments”, The 24th International DMS Conference on Visualization and Visual Languages, DMSVIVA 2018, Redwood City, San Francisco Bay, California, USA, June 2018.

[17] M. Azzari, C. Garau, P. Nesi, M. Paolucci, P. Zamperlin, ”Smart City Governance Strategies to better move towards a Smart Urbanism”, The 18th International Conference on Computational Science and Its Applications, ICCSA 2018. July 2018, Melbourne, Australia.

[18] C. Badii, P.Bellini, D.Cenni, A. Difino, P. Nesi, M. Paolucci, ”User Engagement Engine for Smart City Strategies”, IEEE International Conference on Smart Computing, IEEE SMARCOMP 2017, Hong Kong.

[19] C. Badii, P. Bellini, P. Nesi, M., Paolucci, A smart city development kit for designing Web and mobile Apps, 2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation, 10.1109/UIC-ATC.2017.8397569.