Knowl Inf Syst (2017) 53:699–722
DOI 10.1007/s10115-017-1051-3

REGULAR PAPER

CITIESData: a smart city data management framework

Xiufeng Liu[1] · **Alfred Heller**[2] · **Per Sieverts Nielsen**[3]

Received: 12 August 2016 / Revised: 24 March 2017 / Accepted: 5 April 2017 /
Published online: 12 April 2017

Abstract Smart city data come from heterogeneous sources including various types of the Internet of Things such as traffic, weather, pollution, noise, and portable devices. They are characterized with diverse quality issues and with different types of sensitive information. This makes data processing and publishing challenging. In this paper, we propose a framework to streamline smart city data management, including data collection, cleansing, anonymization, and publishing. The paper classifies smart city data in sensitive, quasi-sensitive, and open/public levels and then suggests different strategies to process and publish the data within these categories. The paper evaluates the framework using a real-world smart city data set, and the results verify its effectiveness and efficiency. The framework can be a generic solution to manage smart city data.

Keywords Data framework · Smart cities · Data privacy · Data quality · Data sensitivity

1 Introduction

1.1 Contextualization

The term "smart cities" dates back to 1980s when there was no common definition agreed globally [10]. Modern cities with a wide use of information and communication technologies

✉ Xiufeng Liu
xiuli@dtu.dk

Alfred Heller
alfh@byg.dtu.dk

Per Sieverts Nielsen
pernn@dtu.dk

1 Danmarks Tekniske Universitet, Building 426, Room 125, 2800 Kgs., Lyngby, Denmark

2 Danmarks Tekniske Universitet, Building 118, Room 211, 2800 Kgs., Lyngby, Denmark

3 Danmarks Tekniske Universitet, Building 426, Room 122, 2800 Kgs., Lyngby, Denmark

(ICT) are coined as smart cities. More recent introduction of the IoT technologies, characterized by the applications of technology, physical and digital connectivity, and advanced algorithms [38], has further supported the technical focus of defining smart cities. The digitized smart cities produce various types of data, which are often transformed for facilitating discovery and accession. This process is known as *data publishing*. The published data are then made accessible via query and/or publish/subscribe facilities [2]. Over the accessing interfaces, the data are eventually integrated into higher-level services or applications that in return enhance the quality and performance of smart city services.

The increased production of data (including personally related data) has also raised an increasing focus on privacy protection. Sharing these data sets across smart cities becomes a challenge, and therefore publishing, the data sets without breaking privacy regulations calls for innovative approaches. For example, according to the policy of information disclosure, data can be categorized into two classes, non-sensitive data, and sensitive data. Non-sensitive data can be treated as open data published without copyright restriction or privacy issue. The data within this category could be governmental documents about public policies, urban or environment development plans, city transportation, population, etc., which are the area closely related to the life of citizens. ICT solutions of managing open data require that the data should be formatted and made accessible so that citizens themselves can reuse the data at their discretion, or companies can use the data and data access to create new, innovative services or applications [21]. Non-sensitive data can be processed according to open standards in order to make the data discovered, accessed, and shared easily. Linked data and linked open data[1] are often regarded as the de facto open standards for the data fusion in smart cities. Data in the other category, sensitive data, are those usually comprising confidential information. The data access to sensitive data is usually strongly restricted by law because the access to the data can result in compromising a person, a system, an application, or other business functions. The access can only be granted to people with personal authorization or those with a business need to know, such as police, and civil administrators. Therefore, a mechanism is needed to model privacy requirements and user privacy preferences so that privacy policies can be enforced in a data platform. For example, the well-known role-based privacy model defines exclusive roles [13] where each role is assigned a minimum set of permissions to perform specific tasks.

Smart city data are published for applications or city services to create new opportunities of knowledge-based decision making. The data are collected from a variety of sources, which are often very diverse and complex, e.g., having different types, formats, meanings, and sizes. Furthermore, smart city data usually have a lot of quality issues, e.g., lacking consistency, completeness, correctness, and accuracy. At the same time data, quality is crucial to provide correct information for decision making. In other words, bad quality data may lead to incorrect analytic results, wrong decision making, with potentially disastrous results. Although there exist many platforms which publish data without processing procedures, quality KPIs, and quality controls, we argue that a robust data platform should provide data with a certain degree of quality guarantee, e.g., publish well-prepared data sets, or otherwise provide some indicators of showing data quality issues.

The objective of this paper is to propose a novel framework for smart city data management where we prioritize data quality assurance and privacy protection in smart city data publishing. We propose some innovative methods and have developed the necessary tools for streamlining smart city data publishing. The paper presents a smart city data management framework, called *CITIESData*. The framework aims at being a generic solution for smart city data

[1] http://www.w3.org/DesignIssues/LinkedData.

fusion and management. To protect data privacy, the proposed approach first classifies the data according to a three-level sensitivity model, i.e., sensitive, quasi-sensitive, and open data (sensitive means the data have the attributes that can directly identify an individual, e.g., social security number; while quasi-sensitive means that the attributes can identify an individual when linking to the external sources such as social network, which include age, gender, address). According to these classifications, different strategies are applied to manage sharing and publishing of the data with different sensitivity levels. To manage the data quality, this paper implements an Big Extraction-Transformation-Loading platform (BigETL) for cleansing large data sets and implements a regression model for grading data quality, as the indicator for labeling data quality for user reference. The proposed data framework can be applied, tailored, or extended to manage different types of smart city data.

In summary, the paper makes the following contributions: (1) proposes a smart city data management framework that is able to share/publish data with different sensitivity levels; (2) implements the framework to streamline data processing, as well as implements the software packages for data quality checking and data anonymization; (3) proposes a regression-based quality checking model according to the features extracted from a data set; (4) evaluates the proposed framework, and validates its effectiveness and efficiency in managing diverse smart city data.

The remainder of the paper is structured as follows. Section 2 summarizes the related work; Sect. 3 presents the architecture of the proposed framework; Sect. 4 describes the implementation, including an ETL program, a data quality checking model, data cleansing, anonymization and publishing; Sect. 5 evaluates the framework; and Sect. 6 concludes the paper with the directions for future work.

2 Related work

Smart cities mainly focus on applying information and communication technologies to all aspects of the life of cities; therefore, the smart city is closely related to ICT concepts, including Big Data, Internet of Things, and the next generation of Internet [32]. Smart cities can make an intelligent response to different kinds of needs, such as public safety and city services, environmental protection, social activities, and improved daily livelihood of the citizens [25]. A Smart ICT platform is the core of the interaction between different stakeholders in a smart city. It is responsible for, among other things, mediating the interaction between data owners/publishers and applications/services [23].

2.1 Smart city data and quality issues

Smart city data come from heterogeneous sources, e.g., various types of Internet of Things, such as traffic, weather, pollution, and noise. Gubbi et al. [11] summarizes smart city data of different domains, including smart homes, cities, retail, transportation, energy, and environments. Payam et al. [1] consider the temporal and spatial characteristics of smart city data, which represent the activities related to cities and people changing over time and locations. They also point out the challenges of handling smart city data and propose a lightweight semantic model which combines data interpolation and analytic models to create data quality parameters, but no further details are given in their work. Data quality has attracted much research interest, e.g., [4,5,24,27,30,36,39]. The authors [4,27,36] tend to classify data quality issues into multiple dimensions, including accuracy, completeness, consistency, timeliness, interpretability, and accessibility, but they have not discussed how to qualify data quality quantitatively. In this paper, we highlight quality issues of smart city data and propose a regression model to quantify data quality, aiming at providing a reference to users before using the data.

2.2 Privacy-preserving data publishing

Privacy-preserving data publishing has received a considerable attention in the past decades, e.g., [6,20,22,33]. There are vast amounts of work on privacy models and anonymization techniques. The k-anonymity model [33] is one of the earliest privacy-preserving data publishing models. In a k-anonymous table, each record is indistinguishable from at least $k-1$ other records with respect to their quasi-identifier (QI) values. Several enhancements of k-anonymity models, including l-diversity [19], t-closeness[14], and (α, k)-anonymity[37], were proposed to provide stronger privacy guarantee. Generalization and suppression are the two most popular anonymization methods [33]. Generalization is applied on the quasi-identifiers and replaces a QI value with a "less-specific but the semantically consistent value" [15], while suppression is to hide a QI value entirely. In our framework, we mainly focus on protecting privacy and securing data management, although we also use both anonymization techniques to protect sensitive information. The privacy protection is implemented through the design of the data platform where different sharing or publishing strategies are used according to data sensitivity levels.

2.3 Data publishing and platforms

Many cities have made their own governmental data publicly available through data portals, such as New York City,[2] London,[3] San Francisco,[4] and Dublin,[5] and the Danish cities, Aarhus,[6] Aalborg,[7] Odense[8] and Copenhagen.[9] Many of these portals are built on top of an open source data platform, CKAN,[10] which operates as a comprehensive data repository for the data from a plethora of government agencies. CKAN is widely regarded as a standardized solution disseminating open data. Besides, various data platforms have been presented in the research literature. For example, Lopez et al. [18] presents a linked data platform, *QuerioCity*, to publish, search, and link city data from static data sets or stream data from sensors. Gao et al. [9] proposes a data platform, *AECIS*, to automate data discovery and urban stream data integration. Santos et al. [29] presents a streamlined process to collect, store, and disseminate monitored data in an urban environment. Bischof et al. [3] presents a *City Data Pipeline* to seamlessly access the data from other data providers. The application, *VIVO*,[11] populates research data, including researcher interests, activities, and accomplishments, which enables researchers to discover other researchers and collaborators across institutions. Darari and Manurung [7] presents a linked data platform, *LinkedLab*, to

[2] https://nycopendata.socrata.com.

[3] http://data.london.gov.uk.

[4] https://data.sfgov.org.

[5] https://data.gov.ie.

[6] http://www.odaa.dk.

[7] http://www.odlaa.dk.

[8] http://odensedataplatform.dk.

[9] http://data.kk.dk.

[10] http://ckan.org.

[11] http://vivoweb.org.

manage the data from research communities. Likewise, our platform aims at managing data publishing services, but furthermore our platform is designed to streamline the whole data process in a secure environment, including collecting, cleansing, anonymization, and data publishing. Moreover, we particularly emphasize data quality issues, privacy protection, and data exchanging. The other works described above are capable of dealing with smart city data, but they are not concerned with data quality and privacy in terms of data sensitivity.

3 Overview of the framework

The primary goal of this framework is to streamline smart city data publishing, while data privacy and data quality are guaranteed. Figure 1 illustrates the system architecture of the proposed framework, which includes the process of data extraction, cleansing, transformation, anonymization, and publishing. The purpose of publishing data is for data discovering, data accession, and data fusion. The following describes the architecture in more detail:

The left-most column in Fig. 1 represents smart city data from heterogeneous sources, such as the data streams from smart meters and sensors, including various types of IoT data (e.g., traffic, weather, pollution, noise, and energy consumption); data from operational systems or data warehouses; and other types of data.

The middle two columns are the data staging area and data transformation layer, respectively. The data staging area is used to save the data from different source systems temporarily before data transformations. A staging area can be any data storage system, such as operating file system, a database management system, or main memory. The data from source systems are extracted into the staging area for the subsequent data transformations. The transformations may include cleansing the data (e.g., correcting misspellings, resolving domain conflicts, dealing with missing elements, or parsing into standard formats), combining data

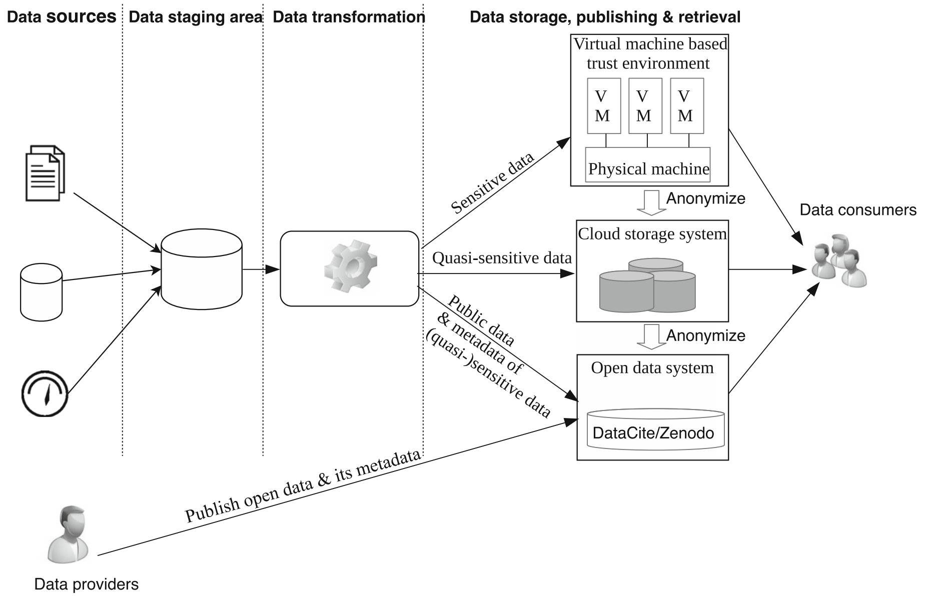


Fig. 1 Architecture of CITIESData framework

from multiple sources, removing duplicate records, and doing data anonymization. These transformations are all prerequisite for the created quality data set to be shared/published.

The right-most column is the data storage, publishing, and retrieval layer. The data in this layer are organized, stored, and made available for retrieval by data consumers, such as citizens, city authorities, business organizations, and applications. The proposed framework publishes data according to a three-level sensitivity model in terms of information disclosure, including sensitive data, quasi-sensitive data, and public (open) data. According to this model, the data are shared or published using different strategies. Sensitive data are shared with authorized users within a virtual machine-based trust environment in the cloud—the data are not allowed to leave this environment when it is used. Quasi-sensitive data are shared via a cloud-based storage system, e.g., a private OwnCloud,[12] whereas public data are published or shared on an open data platform, such as Zenodo,[13] CKAN,[14] or DataCite.[15] Typically, the risk level of the data can be changed through an anonymization process, e.g., sensitive data becomes non-sensitive after being anonymized. An open data platform itself is integrated with a data management system, such as CKAN, which allows data publishers to upload and publish data directly. However, an open data platform can also be restricted to publish metadata (i.e., the data of describing other data). This feature, in fact, is useful for sharing information from (quasi-)sensitive data, i.e., only publishing the metadata while not the (quasi-)sensitive data itself. The benefit is that (quasi-)sensitive data can still be indexed and discovered through the open data platform even though the data themselves are not accessible. If a user needs to access (quasi-)sensitive data, (s)he has to link to a secure environment where user authorization is enforced. The open data portal, thus, becomes a single entrance to search the data available in all data repositories. In this way, the solution becomes a feasible way to maintain privacy and openness of smart city data.

The flowchart in Fig. 2 illustrates the process of accessing the published data. First, a user looks for the desired data via the single visiting entrance, *the data portal*, where the data are organized into different categories. The metadata describe the information about the published data, including the name of the data set, the unique identity, publisher, publisher organization, publishing time, and the link for accessing the data. By following the link, the user can directly access the data if the data are published in the local open data platform, i.e., the data is public. If the data is (quasi-)sensitive data, (s)he has to request the permission of accessing the data by following the link to the private cloud store, where the data are available after the necessary user authorization. If the data are from an external data repository, (s)he may be linked to the external data publishing systems to obtain the data. The metadata in the open data platform are exposed through a standard OAI-PMH protocol such that it can be harvested by other data platforms or search engines, which helps data fusion between smart cities (further details will be given in Sect. 4.5).

4 The implementation

4.1 Requirements of smart city data platform

As mentioned previously, smart cities data come from heterogeneous sources, including various types of Internet of Things data, such as traffic, weather, pollution, and noise data.

[12] https://owncloud.org.

[13] https://zenodo.org.

[14] http://ckan.org.

[15] http://datacite.org.

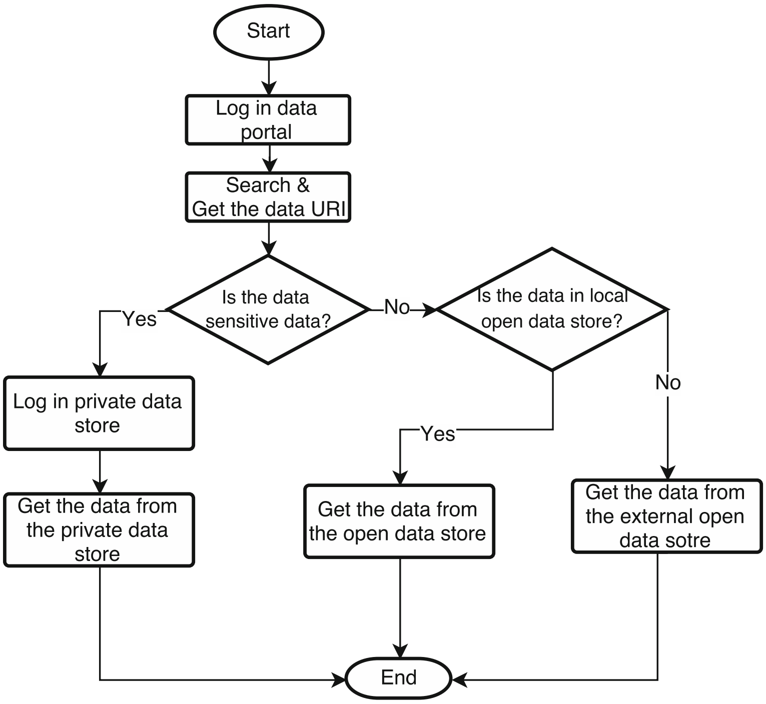

Start
Log in data portal
Search & Get the data URI
Is the data sensitive data?
Yes
No
Is the data in local open data store?
No
Yes
Log in private data store
Get the data from the private data store
Get the data from the open data store
Get the data from the external open data sotre
End

Fig. 2 Flowchart of accessing the published data

Smart city data processing is rather challenging, due to the variety. It involves collecting, preprocessing, storing, analyzing, and visualization. There are many methods for processing and utilizing smart city data. For example, detecting outliers and their removal can be applied to reduce the noise of the data. Distributed storage and data aggregation techniques are applied to conquer the large volume of the data. Furthermore, the data may contain lots of sensitive information since they are highly related to the daily life of citizens. The privacy protection has to be handled accordingly. In this framework, we take these requirements into account and implement a software platform and tools to streamline smart city data processing.

4.2 BigETL: a flexible tool for processing smart city data

Smart city data are featured with Big Data characteristics, such as high volume, variety, velocity, and value (4 Vs). To tackle these challenges, we develop a flexible ETL tool, called *BigETL* [16,17], to deal with scalable big smart city data sets. BigETL is the core component of our CITIESData framework, and it is used for data preprocessing before the data are shared, published, or analyzed. BigETL is open source in the Github repository.[16]

In the following, we discuss how BigETL can meet the requirements for smart city data processing. In Fig. 3, the buildings blocks are shown. The overall system consists of a job scheduler using Quartz,[17] and the user-defined processing modules (algorithms) that are specific to the data processing. BigETL uses multiple data processing systems in its underlying

[16] http://github.org/xiufengliu/bigetl.

[17] http://www.quartz-scheduler.org.

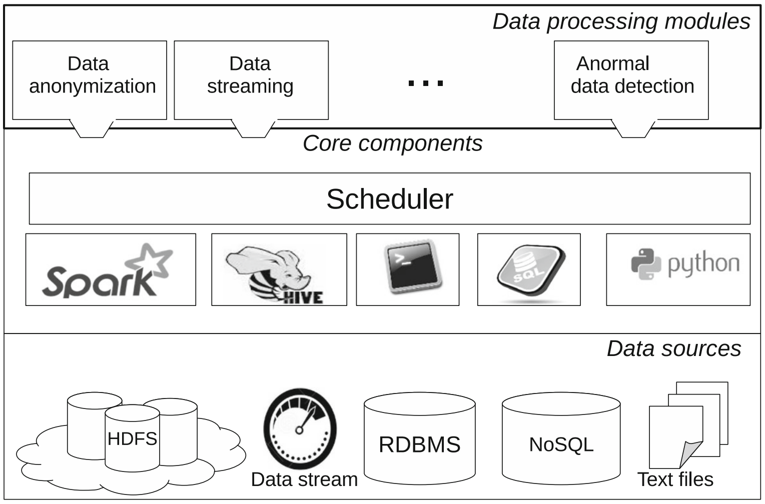


Fig. 3 Building blocks of BigETL

layer. The current version supports Spark, Hive, Linux Shell, SQL Engine, and Python environment, which gives a high flexibility to process different types, formats, and sizes of the data. An algorithm is run directly within the system to achieve a certain processing purpose, such as data cleansing, transformation, anonymization, anomaly detection, data quality scoring. The automatic control of multiple algorithms jointly processing data is through a workflow triggered to run by the scheduler. BigETL supports reading and writing data from/to different source systems, including Hadoop distributed file system (HDFS), smart meters, databases (RDBMS and NoSQL), and text files. A read/write program for a source system is developed by implementing a unique Java programming interface offered by BigETL. Thus, it is easy to implement the read/write program for a new source system.

BigETL can be used to process data streams from IoT networks, e.g., data streams generated by the sensors of monitoring traffic flow, weather condition, air quality, and noise as it supports *Spark Streaming* as its underlying processing system. For processing near real-time data, users have to define the time interval in Spark Streaming to decide how frequent the data should be extracted, e.g., typically every 15 min for smart meter data and every 1 min for capturing traffic flow. Spark Streaming can potentially handle an unlimited size of parallel data streams in the cluster consisting of many nodes, which makes it possible to handle scalable data streams from sensors for citywide deployments. When the data are extracted and read into Spark Streaming, the data are created as the discretized stream (DStream) which represents a continuous data stream received from a source system, or a processed data stream generated by transforming an input stream. DStreams are the fault-tolerant objects partitioned across cluster nodes that can be acted on in parallel. A number of operations, called *transformations*, can be applied to DStreams directly, including map, filter, groupBy, and reduce, as well as windowing operations with user-specified window size and slide interval. Data-dependent programs are implemented for different sources and integrated into BigETL.

BigETL uses Hive as the system for batch processing of large data sets. These data may be from legacy systems, transaction systems, or IoT data accumulated for bulk analytics. It is typically rather challenging to use traditional data warehousing technologies to process the Big data sets, e.g., it may take very long time. Hive is built on top of Hadoop framework [35], which has a good bulk capability. Furthermore, Hive provides a user-friendly command line interface, by which users can implement MapReduce programs by writing SQL-like statements. Its SQL-to-MapReduce translator automatically translates the HiveQL script into the jobs running in a cluster. For smart city data, Hive will be used in the cases where large-scale data sets are analyzed, while fast response time is not required. The results of MapReduce jobs are written into Hive tables, which can be exported to an external database management system for further analysis. Alternatively it can be directly accessed by cloud-based data management systems for data sharing, e.g., by mounting HDFS to OwnCloud. Spark can be another option for batch processing in addition to Hive, but we consider the maturity of HiveQL and its relational-like table organizing the data in HDFS. We have decided to use Hive for batch processing in our implementation.

We have designed BigETL to support several other processing systems to achieve sufficient flexibility for handle diverse smart city data. These other processing systems include Linux Shell, Python, Java virtual machine, and RDBMS, which run in a single-server environment. In this way, users can implement their programs using the different programming languages, i.e., Shell script, SQL, Java, and Python. The programs implemented with different programming languages can be chained to run in a sequential order within a server. This feature is useful for processing the data under complex conditions. BigETL also provides a web-based graphic user interface for facilitating the implementations and provides basic functionality to visualize data in tabular and chart formats, by which users can debug their programs and validate the results.

A data processing program typically consists of multiple processing units, each of which is responsible for a specific task. Figure 13 in “Appendix” shows a typical example of processing smart meter data. This ETL job includes the following tasks: a data transferring program of reading data from a FTP server to a local file system; a program which reads data from local file system to a database, doing in-database data cleansing; a house keeping program of deleting the old data; and a notification program which sends messages to users when the job succeeds and/or fails. All the tasks are organized into a series to fulfill the whole data processing requirements. The joined processing steps are chained to form a workflow. The task for each step runs in a separate underlying processing system. In fact, BigETL supports multiple data workflows for processing the data from different source simultaneously. For example, the platform may run a batch workflow to compute a data model on Hive, while at the same time, another workflow is run to detect anomalies in Spark Streaming. The jobs of workflows are coordinated by a centralized scheduling system, which distributes the jobs according to the available computing resources.

We have implemented two types of scheduling algorithms to control the jobs, including deterministic and un-deterministic. The deterministic scheduling algorithm is used to schedule a job to run at the exact time specified by users. The duration of the job is deterministic and remains the same for repeated executions. The jobs scheduled by this method are typically those that are required to run on a single server. In contrast, the un-deterministic method is for scheduling the jobs to run in a cluster environment. In this case, the jobs may not have to run at the exact time specified by users but typically run at a later time. The purpose is to ensure that there is only a single job running in the cluster at any time. The reason is that Spark or Hive programs run on top of Java virtual machine in a distributed computing environment requiring some amount of memory. If multiple jobs were submitted to the same cluster, it may run out of memory. Our implementation, instead, uses a queue to accommodate all the submitted jobs and sorts them according to the time set by users. If multiple jobs have an identical scheduling time, the scheduler will use the first-in-first-out (FIFO) strategy to submit the jobs. Here, we give up using the built-in job schedulers of Spark and Hadoop (Hive is using Hadoop as its distributed computing framework) in BigETL, which is to improve the flexibility of job management. For example, we can easily chain multiple jobs and run them on the same platform.

4.3 Data preprocessing modules

In CITIESData framework, the goal is to use the data for analytics where data quality is critically important to ensure the correctness of data models and analytic results. Furthermore, smart city data may contain sensitive information that requires desensitization to protect the privacy. We will describe three functional modules for preprocessing the data, including data cleansing, quality checking, and anonymization in the following. These modules are integrated into the BigETL for preprocessing data automatically.

4.3.1 Module 1: data cleansing

One of the major characteristics of smart city data is their variety regarding quality. Data have to be cleansed since poor data quality can lead to unreliable results of any data analytics. Smart city data are from different sources and may have different quality issues. Data quality can be improved through the cleansing process, also called data cleansing or data scrubbing [30]. Data quality is a complex problem. Some quality problems can be easily found using rule-based detection methods, e.g., syntax problems such as the validity of date, while some others are complicated, e.g., semantics problems like data point order in a time series. In our framework, data cleansing can be done automatically by using the rule-based methods, or manually by a visible analytics approach. The automation of data cleansing is as follows: First, define the rules and the dependency in a BigETL's web-based user interface; then define the time interval for running the cleansing job, and finally, the job scheduler will trigger running the cleansing job automatically. Our framework also supports running data cleansing jobs in a cluster if Spark or Hive is used.

Sometimes it is difficult to define the cleansing rules in a computer-executable way for complex data quality issues. The cleansing can be solved in a visual way, i.e., spot the problems visually and fix them, e.g., detect the anomalous values in an energy consumption time series. The surrounding values of an anomaly, combined with consumption patterns is an effective way to judge the suspicious values. By using the proposed ETL tool, it is easy to see the whole picture of data and judge the anomalies quickly. In addition, the data in BigETL can also be visualized through tabular and chart formats including pie, line, bar charts, box and scatter plots, etc. Since BigETL supports a variety of underlying systems including PostgreSQL, Hive, or an in-memory table in Spark, the data can be read from these systems and displayed in different chart formats. Users, therefore, can easily spot the complex data quality issues visually and fix them.

4.3.2 Module 2: data quality checking

It is a good practice to check the data quality before and after the data cleansing. The checking results will provide an indication of data quality and give a reference to users before the data are used. There are two broad data quality checking methods identified, *data profiling* and *data mining* [26]. Data profiling focuses on analyzing values of an individual attribute, e.g., check the data type, length, value range, discrete values and their frequency, variance, uniqueness, occurrence of null values, patterns. The results show the quality of all aspects of the inspected attribute. Data mining is a more advanced method used to detect various data quality related issues. The widely used methods include clustering, summarization, association, and sequence discovery. For example, clustering method can discover the unusual patterns from large energy consumption data set. Functional dependencies between data attributes can be used to discover the integrity of the data and can also be used to fix the data, such as missing values, illegal values, and duplicate values.

We introduce quality scoring after the execution of data cleansing activities for user reference purpose when the data have been published. Our approach uses the data profiling method combined with data mining. The proposed approach uses a multiple linear regression algorithm to score the quality of data derived from [31]. Since it is a supervised machine learning method, we first have to label the data, then use the labeled data to train the linear regression model, and in the end, use the function to grade data quality. We give the formal definition of this algorithm in the following. A data instance is assumed to have m attributes, and its quality is determined by n independent attributes ($n \leq m$), i.e., $A_0, A_1, \ldots, A_{n-1}$. A weight is introduced for each determinant attribute when the overall data quality is graded, i.e.,

$$f = \sum_{i=0}^{n-1} \omega_i * y_i, \quad \omega_0 + \omega_1 + \cdots + \omega_{n-1} = 1.0 \tag{1}$$

where f is the data quality score function, y_i is the quality value of the determinant attribute i, and ω_i is its weight. The quality value of each attribute i is computed by a linear regression function:

$$y_i = \beta_0 + \beta_i x_i^{(1)} + \cdots + \beta_k x_i^{(k)} + \epsilon_i \tag{2}$$

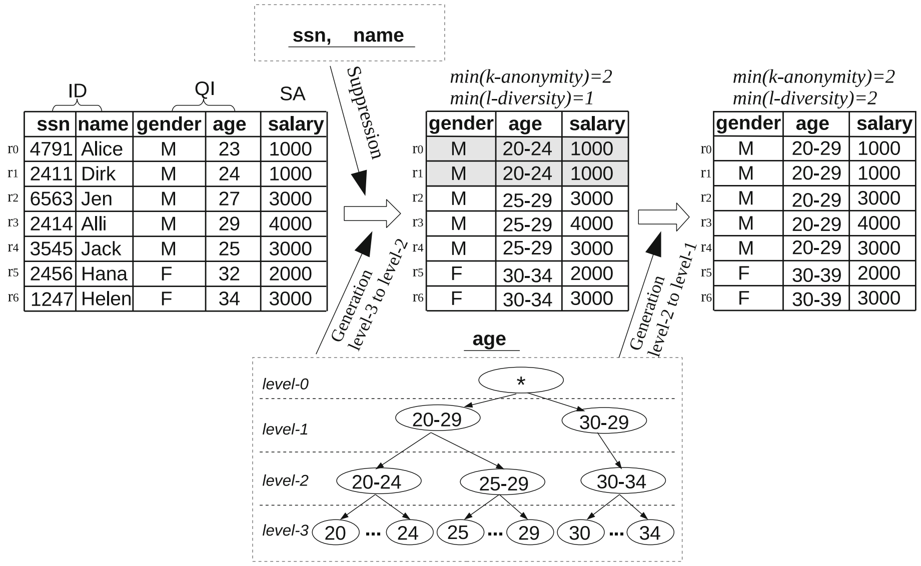
where $x_i^{(1)}, \ldots, x_i^{(k)}$ are the features deciding the quality of the attribute i, $\beta_0, \ldots, \beta_k$ are the coefficient, and ϵ is an error term. An attribute can have different features to determine its quality. For example, for hourly residential electricity consumption, the consumption values typically satisfy a normal distribution. The features of the reading attribute may include the mean, the standard deviation, and the fraction of present readings. Another example is that if the values fit a simple linear regression pattern, the features may include a fraction of the present value, the slope, and the interception. Therefore, the used linear regression model varies to the types of the data being checked. We implement data-dependent programs for checking different data sets, and the quality scores are published as part of the metadata on the data portal. The quality score of each data set is updated for each incremental loading in our data platform.

4.3.3 Module 3: data anonymization

As mentioned earlier, some smart city data may contain sensitive information. These data are typically the person-related data collected from the sources, including banks, government organizations (e.g., tax and citizen offices), hospitals (medical history), utilities, social networks, smart phones, smart meters, and other types of ubiquitous sensors. Since publishing such data immediately violates personal related privacy, they have to be desensitized through a sequence of anonymization operations. Anonymization refers to the approaches that seek to hide the identity and the sensitive data of record owners [8]. Formally, a table of T containing sensitive attributes can be defined in a 4-tuple format, i.e.,

$$T =< ID, QI, SA, EA > \tag{3}$$

where ID is a non-empty set of attributes that can explicitly identify record owners, such as name and social security number (SSN); QI is a non-empty set of the attributes that can potentially identify record owners by linking the external information, such as gender, age,



	ssn	name	gender	age	salary
r0	4791	Alice	M	23	1000
r1	2411	Dirk	M	24	1000
r2	6563	Jen	M	27	3000
r3	2414	Alli	M	29	4000
r4	3545	Jack	M	25	3000
r5	2456	Hana	F	32	2000
r6	1247	Helen	F	34	3000

	gender	age	salary
r0	M	20-24	1000
r1	M	20-24	1000
r2	M	25-29	3000
r3	M	25-29	4000
r4	M	25-29	3000
r5	F	30-34	2000
r6	F	30-34	3000

	gender	age	salary
r0	M	20-29	1000
r1	M	20-29	1000
r2	M	20-29	3000
r3	M	20-29	4000
r4	M	20-29	3000
r5	F	30-39	2000
r6	F	30-39	3000

Fig. 4 An example of data anonymization implementation with the requirements, $k\text{-}anonymity \geq 2$ and $l\text{-}diversity \geq 2$

and home address; SA is a non-empty set of attributes that describes person-specific sensitive information such as disease, salary, and disability status; and EA is a set of other attributes that do not fall into the previous three categories.

There are two often-used metrics to indicate the anonymity of a data set, namely k-*anonymity* and l-*diversity* [28]. If a table satisfies k-anonymity, every record in the table is indistinguishable from at least $k - 1$ other records on every set of QI attributes. This ensures that individuals cannot be uniquely identified by linking attacks. l-*diversity* means that for a given QI value, there are at least l distinct values in the SA attribute. This is to avoid homogeneity or background knowledge attack. There are two techniques to achieving the anonymity requirement, i.e., *suppression* and *generalization* [33]. For a better understanding, we now explain them by the example shown in Fig. 4, which is to publish employee data with the anonymity requirement of $k\text{-}anonymity \geq 2$ and $l\text{-}diversity \geq 2$. The following anonymization steps are conducted. First, suppression is employed to hide the individual identifiers, i.e., hiding the values of *ssn* and *name* attributes. Second, generation is applied on the QI attributes to achieve k-anonymity ($k = 2$ in this example). A hierarchy tree is built for the generalization on the *age* attribute, which has the height of four. If the age is generalized from *level*-3 to *level*-2, three equivalent anonymous groups are generated according to the QI attribute values, i.e., $\{r_0, r_1\}$, $\{r_2, r_3, r_4\}$ and $\{r_5, r_6\}$ (see the table in the middle in Fig. 4). The anonymity of the resulting groups is ≥ 2, but the minimum diversity of the groups is 1 since the rows r_0 and r_1 in the first group have identical values of the SA attribute, i.e., $salary = 1000$. Therefore, the values in the column *age* have to be further generalized to *level*-1; then, the results satisfy the requirement (see the table on the right in Fig. 4).

We have developed a software package for facilitating the anonymization process. The implementation of this example is shown in Fig. 5. The code snippet is self-explanatory, including defining the hierarchy trees of age and gender attribute values (see line 4–15), the attributes types (see line 17–22), the anonymizer, and the anonymization requirements (see

```
1  Data data = Data.create("input.csv");
2
3  // Define hierarchies
4  final DefaultHierarchy age = Hierarchy.create();
5  age.add("23", "20-24", "20-29", "*");
6  age.add("24", "20-24", "20-29", "*");
7  age.add("27", "25-29", "20-29", "*");
8  age.add("29", "25-29", "20-29", "*");
9  age.add("25", "20-24", "20-29", "*");
10 age.add("32", "30-34", "30-39", "*");
11 age.add("34", "30-34", "30-39", "*");
12
13 final DefaultHierarchy gender = Hierarchy.create();
14 gender.add("male", "*");
15 gender.add("female", "*");
16
17 // Set data attribute types
18 data.getDefinition().setAttributeType("ssn", AttributeType.ID);
19 data.getDefinition().setAttributeType("name", AttributeType.ID);
20 data.getDefinition().setAttributeType("age", age);
21 data.getDefinition().setAttributeType("gender", gender);
22 data.getDefinition().setAttributeType("salary", AttributeType.SA);
23
24 // Define privacy requirements
25 final Anonymizer anonymizer = new Anonymizer();
26 final Configuration config = Configuration.create();
27 config.addCriterion(new Kanonymity(2));
28 config.addCriterion(new Ldiversity(2));
29
30 // Perform anonymization
31 Result result = anonymizer.anonymize(data, config);
32
33 // Write result
34 result.getHandle().write("output.csv");
```

Fig. 5 Example of anonymization implementation

line 24–28). The software package is distributed as a library in CITIESData platform. The implementation of anonymization can be run by the BigETL tool to anonymize sensitive data automatically by running the jobs.

4.4 Data sharing/publishing strategies

After the data are preprocessed, the data are loaded into the data management system for publication. According to the degree of information disclosure, we divide the data into three risk levels, including sensitive, quasi-sensitive, and open/open data. Each risk level of data will be published or shared using the different strategies discussed below.

4.4.1 Sensitive data

Sensitive data are the value from the quasi-identifier attribute QI, such as personal age, address, and properties. Sensitive data can be used to identify an individual by linking to external information. However, the value of the identifier attribute ID explicitly identifies an individual, such as the name and social security number. Therefore, these values will never be shared or published on our data platform. The ID attribute value is anonymized by the suppression technique.

If analytical work requires use of sensitive data, which is the case in some data modeling, we use a virtual machine (VM)-based approach to protect the use of sensitive data. In our framework, we employ OpenNebula,[18] an open source cloud computing infrastructure management system, to create a *trust execution environment* for data users. The execution environment is located within the virtual machine where users can run their applications using all risk-level data. The virtual machine is created from a custom image with administrative privileges disabled and preinstalled the applications for data modeling or analysis. The root of the trust environment is the hardware component on the motherboard, called the *Trusted Platform Module (TPM)*, which is enabled to encrypt the data on the local hard drive. If a user needs to use a trust execution environment, (s)he must create a virtual machine through OpenNebula, which will find the appropriate computing resource to create the VM instance automatically through its resource scheduling system. The environment is equipped with TPM and a secure Xen hypervisor containing a Transparent Memory Encryption Component (TMEC). The secure Xen hypervisor controls interdomain isolation to ensure that jobs that run on different virtual machines do not interact with each other. Sensitive data and VM images are saved on a secure hard disk drive, and the results of user applications are written to the same hard disk drive.

4.4.2 Quasi-sensitive data

Quasi-sensitive data refer to the values of SA attributes. Quasi-sensitive data can be linked to sensitive data to expose individuals, such as energy consumption data and employee salaries. Quasi-sensitive data, however, are much less revealing if they are not linked to sensitive data. For many analysis tasks, quasi-sensitive data are sufficient, such as pattern discovery and clustering based on the found patterns. Thus, quasi-sensitive data can be managed and shared through a much looser environment, such as using the cloud with appropriate user authentication. In our framework, we employ the cloud data management system *OwnCloud*[19] to share quasi-sensitive data. The OwnCloud service of our Smart City Project[20] is provided by the Danish Electronic Infrastructure (DeIC),[21] an organization providing e-infrastructures (computing, storage, and network) to Danish universities. When we wrote this paper, the OwnCloud service is still at the beta version. It is integrated with the Danish user access service[22] to support single sign-on. As a result, all users from the Universities and the Institutes in Denmark can log into the DeIC OwnCloud through their organization's accounts and access the data.

The OwnCloud service is a web application that utilizes a standard LAMP stack (Linux, Apache, MySQL, PHP). All data in the OwnCloud are stored in the file system of the application server, but the meta-information of the data and users is saved to the underlying MySQL database, such as the information of files, the folder, shared properties, user preferences, and access permissions. There are multiple ways to administrate the data in OwnCloud, including

[18] http://opennebula.org.

[19] https://owncloud.org.

[20] http://smart-cities-centre.org.

[21] http://www.deic.dk.

[22] http://www.wayf.dk.

Table 1 Permission management in OwnCloud

Group-members ProjectGroup	Member
CITIES	John; Alice; Joe, Mike
SmartHome	John; Rose;
iParking	Smith; Liu

Access permission Data set	CITIES			SmartHome			iParking		
	Read	Write	Share	Read	Write	Share	Read	Write	Share
District heating data	√	√	√	√	√		√	√	
Electricity cons, data	√	√	√	√	√		√	√	√
...	…			…			…		

the web interface, the WebDAV[23] interface, or sync-clients for desktop and mobile phone. The interface supports various file operations, including upload, download, delete, rename, and share. In the OwnCloud, the quasi-sensitive data are organized into different categories according to its types, such as district heating data, building data, electricity data. In addition, we implement a fine-grained permission control policy to manage data access permissions at the folder level. We create a user group for each project and assign the membership to the project member (see the top table in Table 1). For each project group, the permissions including read, write, and share are granted (see the bottom table in Table 1).

4.4.3 Public data

Smart cities share a variety of data between citizens, government agencies, and organizations, such as video, code, books, software, and government documents, called *public data* or *open data*. The wide range of data definitions is not just limited to data sets; instead, it includes methods, source code, models and packages, and many others. All data ranges should be able to be shared, for example, through a common data platform. CITIESData uses *zenodo* as the public data platform to publish data. Zenodo was developed by CERN at the OpenAIREplus project.[24] The reason we chose Zenodo as our open data platform is that it natively supports the release of various data in different areas. Each data set published in Zenodo will be assigned a unique digital object identifier (DOI) through which the data set can be referenced like an academic article. Our platform publishes the data from many researchers, including raw data sets, and their research results such as scientific papers, code and models. The Zenodo platform enables the data and work to be discovered and cited by other researchers easily.

4.5 Metadata and data fusion

All of the data managed in our platform are available to the public for searching and indexing (for the (quasi-)sensitive data, only the metadata are published in the open data platform).

[23] https://tools.ietf.org/html/rfc4918.

[24] http://www.openaire.eu.

When the data are published, some necessary information has to be provided, including the title, data author, organization, and others. When the data are submitted, the system automatically generates the metadata file according to the standard of DataCite Metadata Schema,[25] which is accepted by the Zenodo data platform. Metadata refer to the descriptors or tags that identify a document, data set, data model, or software. CITIESData will be extended in our future work to support other schema standards.

Data fusion is the main motivation to build this smart city data management framework. Data fusion plays a central role in the inner- and intra-cities communication. The metadata of all the data published in our data platform are exposed via OAI-PMH protocol natively supported by Zenodo. OAI-PMH is an open standard for exchanging structured metadata through URIs, HTTP, and XML [12]. In Zenodo, metadata can also be exported in several other standard formats, including MARCXML, Dublin Core, and DataCite Metadata Schema. As a result, metadata can be discovered and harvested by other data service providers or search engines. This is crucial since it is the way to link isolated data, which helps data fusion in smart cities.

5 Evaluation

We now evaluate the effectiveness and efficiency of our framework in publishing data. The system is installed in the server, with an Intel(R) Core(TM) i7-4770 processor (3.40 GHz, 4 Cores, hyper-threading is enabled, two hyper-threads per core), 16 GB RAM, and a Seagate Hard driver (1TB, 6 GB/s, 32 MB Cache and 7200 RPM), running Ubuntu 12.04 LTS with 64bit Linux 3.11.0 kernel. We use the DeIC's OwnCloud service to store data and use Zenodo as an open data platform. In our experiments, we used in-database data cleansing method to cleanse the data which used PostgreSQL 9.4. PostgreSQL has the following settings: "shared buffers = 2048 MB, temp buffers = 256 MB, work mem = 512 MB, checkpoint segments = 64" and the default values for other parameters.

We use a real-world district heating data set for our evaluation. The original data set contains hourly heating consumption readings of 53,000 households with a length of fourteen months, and the customer information of all the households.

5.1 Effectiveness

We first evaluate the effectiveness of data transformation and data anonymization. A sample of the raw data set is shown on the left side of Fig. 6, which consists of customer data and heating consumption data. The customer data contain sensitive information, including the customer names and addresses (road and building number), while the heating consumption data are time series data with two metrics, *volume* and *heat energy*, and read time (we omit showing other attributes for space reason). The heating consumption data have some data quality issues, including duplicate rows and missing values for certain time series. The data in both tables will be published according to the criteria of privacy and quality.

We follow the steps listed in Fig. 7 to process and publish the data. The CITIESData platform provides a web-based user interface to implement ETL programs (see Fig. 13 in "Appendix"). On each day, data from the data source are uploaded to the staging area, *a ftp server*. The data are then read and processed by the ETL program running on BigETL. We use the database management system PostgreSQL to do in-database data cleansing,

[25] http://schema.datacite.org/meta/kernel-3.

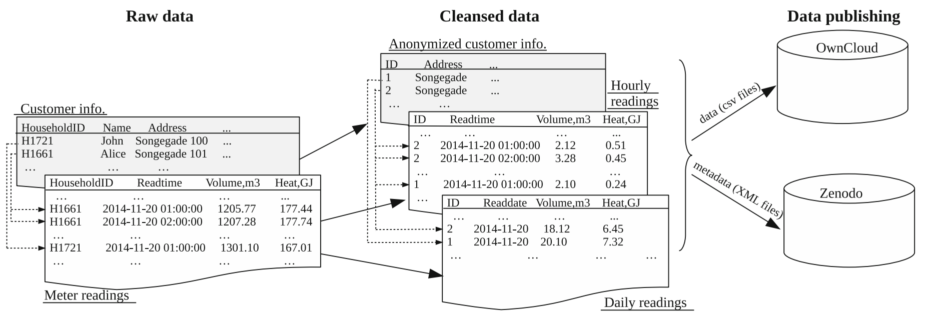

Raw data
Cleansed data
Data publishing
Customer info.
HouseholdID Name Address ...
H1721 John Songegade 100 ...
H1661 Alice Songegade 101 ...
HouseholdID Readtime Volume,m3 Heat,GJ
H1661 2014-11-20 01:00:00 1205.77 177.44
H1661 2014-11-20 02:00:00 1207.28 177.74
H1721 2014-11-20 01:00:00 1301.10 167.01
Meter readings
Anonymized customer info.
ID Address ...
1 Songegade ...
2 Songegade ...
Hourly readings
ID Readtime Volume,m3 Heat,GJ
2 2014-11-20 01:00:00 2.12 0.51
2 2014-11-20 02:00:00 3.28 0.45
1 2014-11-20 01:00:00 2.10 0.24
ID Readdate Volume,m3 Heat,GJ
2 2014-11-20 18.12 6.45
1 2014-11-20 20.10 7.32
Daily readings
data (csv files)
metadata (XML files)
OwnCloud
Zenodo

Fig. 6 Example of data transformation and anonymization

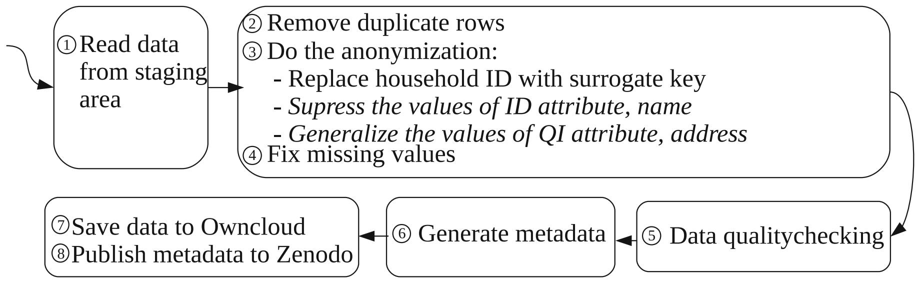

① Read data from staging area
② Remove duplicate rows
③ Do the anonymization:
- Replace household ID with surrogate key
- *Supress the values of ID attribute, name*
- *Generalize the values of QI attribute, address*
④ Fix missing values
⑤ Data qualitychecking
⑥ Generate metadata
⑦ Save data to Owncloud
⑧ Publish metadata to Zenodo

Fig. 7 Process of publishing the heating data

i.e., the data are first loaded into PostgreSQL, and then the cleansing is conducted in the database. The cleansing algorithm is implemented as a store procedure through the Pl/pg SQL scripting language. The cleansing involves looking up the consumption data table using the business keys, `HouseholdID` and `Readtime`, to identify duplicate records (rows with smaller timestamps are discarded). Since the measures, `volume` and `heat energy`, are accumulative readings, the actual consumption has to be calculated by the readings of two continuous hours. For the anonymization of customer data, the processing operations include generating new surrogate keys for replacing the original household IDs, suppressing the customer name by filtering, and generalizing the addresses by suppressing the building number.

Data quality checking comes after the cleansing for the indication purpose when the data are published. We first generate the multivariate linear regression model according to Eq. 2, which is done through the training process. We use semiannual time series as the training data set and use the rest for verification. For the district heating data set, there are two deterministic attributes, `volume` and `heatEnergy`. However, the attribute `heatEnergy` depends on `volume`, i.e., the heat energy can be computed by the used volume. Therefore, we choose `volume` attribute for our study. We use the mean, standard deviation, and the absent ratio of the values as the features to train the model. We deliberately omit some of the attribute values in the consumption time series and label the quality manually for the training data set (shown in the table on the left and on the right, respectively, in Table 2). We grade the data based on our knowledge of the data used in our research work, i.e., we use the data for comparing the heat consumption change of a building after a major renovation. The presence of the values is our primary consideration when we grade the quality of this data set. Here, we use the simple formula, $100 * (1 - absentRatio)$, to calculate the score. In the common use cases,

Table 2 Preparation of the training data sets

Consumption time series		Data quality scoring			
ID	UsageTimeseries	Mean	SD	AbsentRatio	AbsentRatio
1	0.11,0.13,,0.09,0.12,…	0.14	6.21	0.08	92
2	1.1,1.0,0.98,0.911.2,…	1.14	0.47	0.02	98
3	0.9,1.14,0.99,1.10,1.2,.	1.27	1.82	0.03	97
4	0.21,0.51,,,0.34,0.5,…	0.35	8.21	0.11	89
…		…			

Table 3 Coefficients and the validity of the multiple linear regression model

Explanatory variable	Coefficient estimate	SE	t value	Two-tailed p test	Significance
Intercept	94.139	0.0729	7.21	1.24e−10	***
Mean	2.33	0.0325	4.51	0.0412	*
stddev	−0.0533	0.0287	3.01	0.026	*
AbsentRatio	−22.159	0.0814	8.21	2.08e−11	***

0 '***', 0.001 '**', 0.01 '*', 0.05 '.' 0.1', Adjusted R^2: 0.714, n=53,000

this process is usually carried out by a data expert with domain knowledge. We select the features including the mean, standard deviation, and the absent ratio for the linear regression model, i.e., $f = \beta_0 + \beta_1 * mean + \beta_2 * stddev + \beta_3 * absentRatio$. Table 3 shows the coefficients and their validity generated by the training process. The number of "*" indicates the significance level of each explanatory variables. According to the results, the coefficient estimate of absent ratio has the most significance, as well as the intercept which is a constant in the linear regression model. t and p values are the statistics values of showing the evidence of the significant difference between population mean and a hypothesized value. t and p are inextricably linked. A bigger t value represents the more significance.

We now use the trained model to grade the data quality for all the time series. Figure 8 displays the distribution of data quality scores which fall within 80 to 98. Among of them, 46,004 time series are within 90 and 98, accounting for 86.8%.

In the end, we generate the metadata XML file according to the standard of DataCite Metadata Schema version 3.1. The contents of the XML file include the DOI for identifying the data set, the information of publisher, the data quality information in the description, and others (see Fig. 9). The file is uploaded and published on the Zenodo open data platform.

5.2 Efficiency

We now evaluate the efficiency by comparing two types of data cleansing methods. The first method is doing data cleansing by *ETL* using pygrametl [34], a Python programming-based ETL tool. The ETL method does data transformation before the loading. The second method is called *ELT*, which conducts in-place transformation in the data warehouse after the loading. In this method, we use PostgreSQL stored procedures to cleanse the data, which are implemented using Pl/pgSQL programming language. The two methods are scheduled

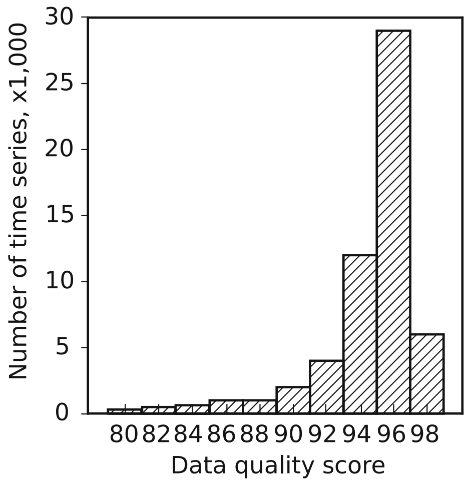


Fig. 8 Data quality score distribution

```
<?xml version="1.0" encoding="UTF-8"?>
<resource xsi:schemaLocation="" xmlns="http://datacite.org/schema/kernel-3"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <identifier identifierType="DOI">dx.doi.org/10.5281/zenodo.31503</identifier>
  <creators>
    <creator>
      <creatorName>Xiufeng Liu</creatorName>
      <nameIdentifier nameIdentifierScheme="" schemeURI=""></nameIdentifier>
      <affiliation xmlns="">Technical University of Denmark</affiliation>
    </creator>
  </creators>
  <titles>
    <title>District heating hourly data set</title>
  </titles>
  <publisher>Xiufeng Liu</publisher>
  <publicationYear>2015</publicationYear>
  <description>This data set contains 53k time series of a hourly resolution.
 46k time series have the score between 90 and 98, accounting 86.8%.
 </description>
</resource>
```

Fig. 9 Metadata of published data

to run on our BigETL tool using the underlying processing systems of *pygrametl* and *PostgreSQL*. We divide time series into four pieces with a same window size, and the pieces are processed in order according to the time. In both methods, we use PostgreSQL's bulk-load utility, *COPY*, to load data into the data warehouse. Using pygrametl, we do the on-the-fly data cleansing. For each of the rows, the program looks up the target table in the data warehouse to check the existence of the row. The lookup operation is conducted on the business key, `householdID`, and the timestamp, `readtime`, on which b-tree index is created, and clustered index is created on `householdID`. While, for the ELT method, we create the same indexes on these attributes and use the SQL `DISTINCT` operator to filter duplicate rows.

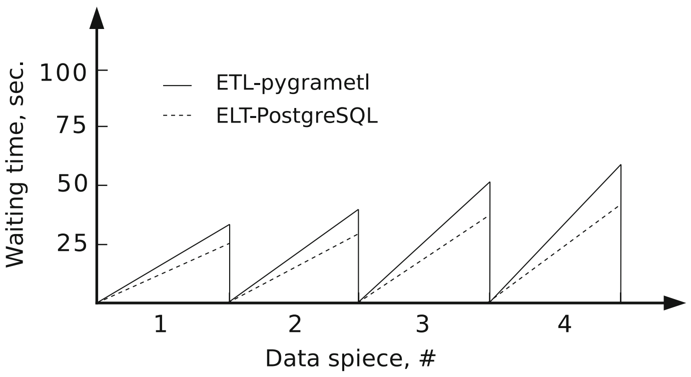
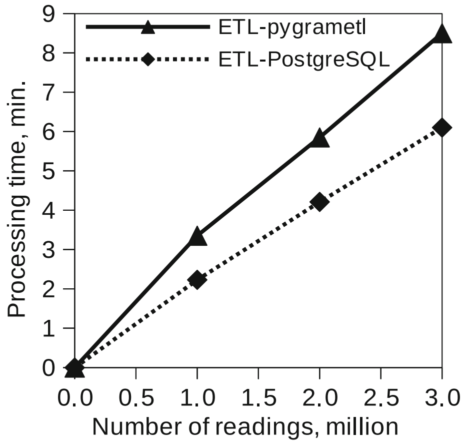


Fig. 10 waiting time of getting the latest published data

Fig. 11 Processing time



We first measure the efficiency from the point of the view of data consumers, i.e., to measure the waiting time of getting the latest published data. Figure 10 shows the waiting times for both methods. From the results, we could observe that the waiting times rise slightly for both methods, mainly due to the increasing size of the data in the target table when verifying the duplicate records. The ELT method can reduce waiting time more than the ETL method since the data can be efficiently processed in the data warehouse.

We now measure the performance by the scale-up experiment. We scale the number of readings from 1 to 5 million and measure the time for both processing methods. Figure 11 shows the processing time, while Fig. 12 shows the throughput (tested with 1 million readings). The processing time of the two scales almost linearly with the increasing number of hourly readings and the time of the ELT shows less than ETL method in overall. The corresponding throughput of ELT is roughly 1.4 times the ETL method. The results have shown that the in-database cleansing method (i.e., ELT) outperforms the classic cleansing method (i.e., ETL). The main reason may be due to fewer data movements using the ELT method since the data have been loaded, and the performance of ELT is also highly dependent on the underlying data management system.

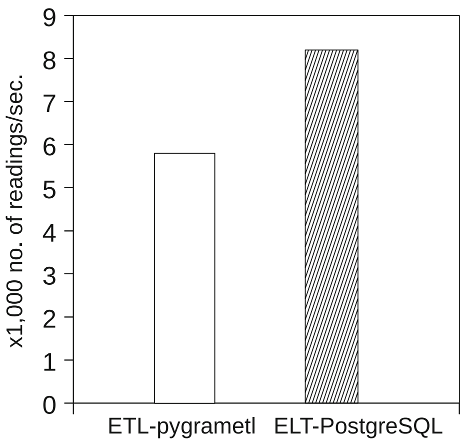


Fig. 12 Throughput

5.3 Remark

There are various use cases for handling smart city data, which is typically challenging to use a uniform data processing system or platform. In the above experiments, we showed an example of how to use our platform to handle smart city data. We used energy data sets to evaluate our platform and validated its effectiveness and efficiency. As we have already mentioned, the proposed framework supports a number of underlying data processing systems. This feature makes our framework distinct to others, i.e., supports processing different types of smart city data from extracting to publishing within the same platform. We believe that the proposed framework is a good candidate for a common solution for smart city data management.

6 Conclusion and future work

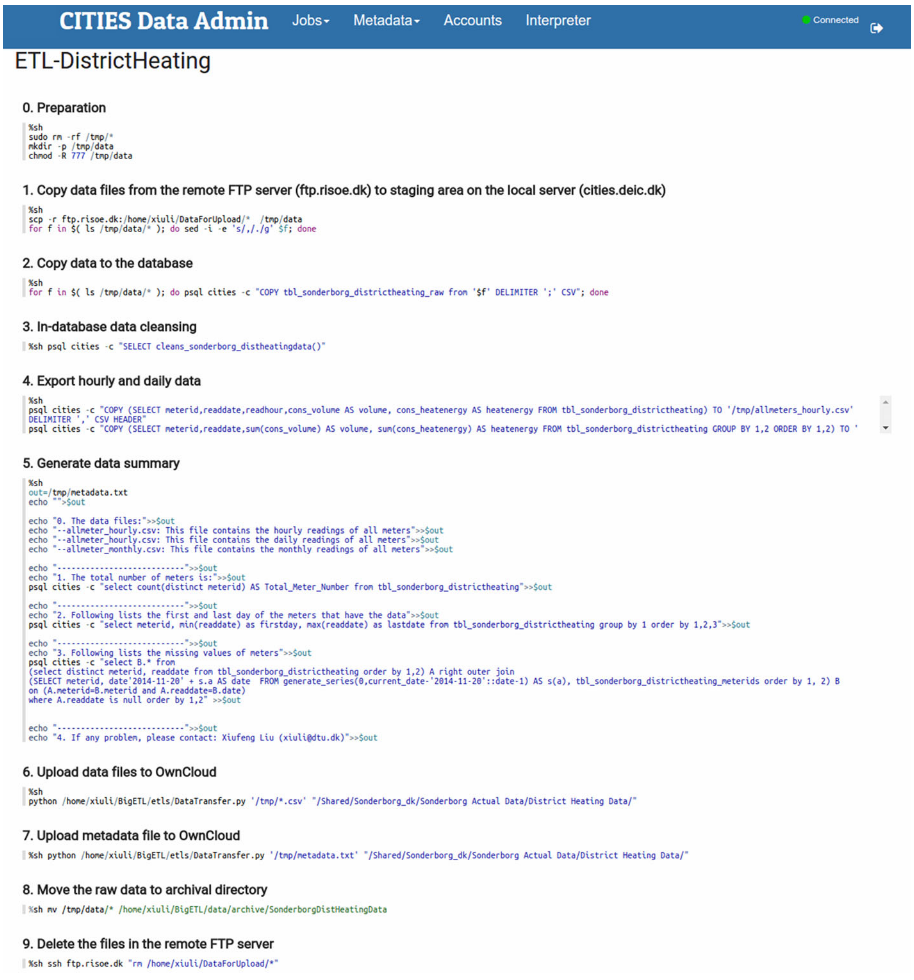
Smart cities generate various data that need to be shared or published for different purposes. In this paper, we have proposed the Smart City Data Management Framework, *CITIESData*. We have proposed a three-risk-level model and categorized smart city data into sensitive data, semi-sensitive data, and public data, which are published or shared by different strategies. In order to facilitate smart city data management, we have developed a flexible and powerful ETL tool, *BigETL*, for big data processing. We also implemented a linear regression algorithm and a library for data quality checking and anonymization. We have evaluated the proposed data framework based on a real-world smart city data set, and the results have shown the effectiveness and efficiency of using the framework.

In the future, first we will extend our BigETL tool to support more other data processing systems in order to provide more flexibility and capability for processing diverse smart city data. Second, the libraries for handling different data issues will be developed, such as anomaly detection and data quality. Third, we will extend to support other schema standards for the metadata management.

Acknowledgements This research was supported by the CITIES Project (No. 1035-00027B) funded by Innovation Fund Denmark. The infrastructure components are partly supported by the Danish Electronic Infrastructure (DeIC) through the project "Science Cloud for Cities."

Appendix

See Fig. 13.



ETL-DistrictHeating

0. Preparation

```
%sh
sudo rm -rf /tmp/*
mkdir -p /tmp/data
chmod -R 777 /tmp/data
```

1. Copy data files from the remote FTP server (ftp.risoe.dk) to staging area on the local server (cities.deic.dk)

```
%sh
scp -r ftp.risoe.dk:/home/xiuli/DataForUpload/*  /tmp/data
for f in $( ls /tmp/data/* ); do sed -i -e 's/,/./g' $f; done
```

2. Copy data to the database

```
%sh
for f in $( ls /tmp/data/* ); do psql cities -c "COPY tbl_sonderborg_districtheating_raw from '$f' DELIMITER ';' CSV"; done
```

3. In-database data cleansing

```
%sh psql cities -c "SELECT cleans_sonderborg_distheatingdata()"
```

4. Export hourly and daily data

```
%sh
psql cities -c "COPY (SELECT meterid,readdate,readhour,cons_volume AS volume, cons_heatenergy AS heatenergy FROM tbl_sonderborg_districtheating) TO '/tmp/allmeters_hourly.csv'
DELIMITER ',' CSV HEADER"
psql cities -c "COPY (SELECT meterid,readdate,sum(cons_volume) AS volume, sum(cons_heatenergy) AS heatenergy FROM tbl_sonderborg_districtheating GROUP BY 1,2 ORDER BY 1,2) TO '
```

5. Generate data summary

```
%sh
out=/tmp/metadata.txt
echo "">$out

echo "0. The data files:">>$out
echo "--allmeter_hourly.csv: This file contains the hourly readings of all meters">>$out
echo "--allmeter_hourly.csv: This file contains the daily readings of all meters">>$out
echo "--allmeter_monthly.csv: This file contains the monthly readings of all meters">>$out

echo "-------------------------">>$out
echo "1. The total number of meters is:">>$out
psql cities -c "select count(distinct meterid) AS Total_Meter_Number from tbl_sonderborg_districtheating">>$out

echo "-------------------------">>$out
echo "2. Following lists the first and last day of the meters that have the data">>$out
psql cities -c "select meterid, min(readdate) as firstday, max(readdate) as lastdate from tbl_sonderborg_districtheating group by 1 order by 1,2,3">>$out

echo "-------------------------">>$out
echo "3. Following lists the missing values of meters">>$out
psql cities -c "select B.* from
(select distinct meterid, readdate from tbl_sonderborg_districtheating order by 1,2) A right outer join
(SELECT meterid, date'2014-11-20' + s.a AS date  FROM generate_series(0,current_date-'2014-11-20'::date-1) AS s(a), tbl_sonderborg_districtheating_meterids order by 1, 2) B
on (A.meterid=B.meterid and A.readdate=B.date)
where A.readdate is null order by 1,2" >>$out


echo "-------------------------">>$out
echo "4. If any problem, please contact: Xiufeng Liu (xiuli@dtu.dk)">>$out
```

6. Upload data files to OwnCloud

```
%sh
python /home/xiuli/BigETL/etls/DataTransfer.py '/tmp/*.csv' "/Shared/Sonderborg_dk/Sonderborg Actual Data/District Heating Data/"
```

7. Upload metadata file to OwnCloud

```
%sh python /home/xiuli/BigETL/etls/DataTransfer.py '/tmp/metadata.txt' "/Shared/Sonderborg_dk/Sonderborg Actual Data/District Heating Data/"
```

8. Move the raw data to archival directory

```
%sh mv /tmp/data/* /home/xiuli/BigETL/data/archive/SonderborgDistHeatingData
```

9. Delete the files in the remote FTP server

```
%sh ssh ftp.risoe.dk "rm /home/xiuli/DataForUpload/*"
```

Fig. 13 Web-based user interface of implementing an ETL program on CITIESData platform

References

1. Barnaghi P, Bermudez-Edo M, Tonjes R (2015) Challenges for quality of data in smart cities. J Data Inf Qual 6(2–3):6
2. Bischof S, Karapantelakis A, Nechifor CS, Sheth A, Mileo A, Barnaghi P (2014) Semantic modelling of smart city data. In: W3C workshop on the web of things—enablers and services for an open web of devices. W3C
3. Bischof S, Polleres A, Sperl S (2013) City data pipeline In: Proceedings of the I-SEMANTICS posters and demonstrations track, p 45

4. Bovee M, Srivastava RP, Mak B (2003) A conceptual framework and belief-function approach to assessing overall information quality. Int J Intell Syst 18(1):51–74
5. Cappiello C, Francalanci C, Pernici B (2003) Time-related factors of data quality in multichannel information systems. J Manag Inf Syst 20(3):71–91
6. Carpineto C, Romano G (2015) Kθ-affinity privacy: releasing infrequent query refinements safely. Inf Process Manag 51(2):74–88
7. Darari F, Manurung R (2011) LinkedLab: a linked data platform for research communities. In: Advanced computer science and information system (ICACSIS), pp 253–258
8. Fung B, Wang K, Chen R, Yu PS (2010) Privacy-preserving data publishing: a survey of recent developments. ACM Comput Surv (CSUR) 42(4):14
9. Gao F, Ali MI, Mileo A (2014) Semantic discovery and Integration of urban data streams In: Proceedings of the 5th workshop on semantics for smarter cities, pp 15–30
10. Glasmeier A, Christopherson S (2015) Thinking about smart cities. Camb J Reg Econ Soc 8(1):3–12
11. Gubbi J, Buyya R, Marusic S, Palaniswami M (2013) Internet of things (IoT): a vision, architectural elements, and future directions. Future Gener Comput Syst 29(7):1645–1660
12. Haslhofer B, Schandl B (2008) The OAI2LOD server: exposing OAI-PMH metadata as linked data. In: Proceedings of WWW workshop linked data on the web
13. He Q, Antón AI (2003) A framework for modeling privacy requirements in role engineering. In: Proceedings of REFSQ, pp 137–146
14. Li N, Li T, Venkatasubramanian S (2007) t-closeness: privacy beyond k-anonymity and l-diversity. In: Proceedings of ICDE, pp 106–115
15. Li T, Li N (2009) On the tradeoff between privacy and utility in data publishing. In: Proceedings of SIGKDD, pp 517–526
16. Liu X, Nielsen PS (2015) Streamlining smart meter data analytics. In: Proceedings of the 10th conference on sustainable development of energy, water and environment systems. SDEWES2015.0558, pp 1–14
17. Liu X, Nielsen PS (2016) An ICT-solution for smart meter data analytics. Energy 115(3):1710–1722
18. Lopez V, Kotoulas S, Sbodio ML, Stephenson M, Gkoulalas-Divanis A, Aonghusa PM (2012) QuerioCity: a linked data platform for urban information management. The semantic web, pp 148–163
19. Machanavajjhala A, Kifer D, Gehrke J, Venkitasubramaniam M (2013) l-diversity: privacy beyond k-anonymity. ACM Trans Knowl Discov Data 1(1):3
20. Malin B (2008) k-unlinkability: a privacy protection model for distributed data. Data Knowl Eng 64(1):294–311
21. Manville C, Cochrane G, Cave J et al (2014) Mapping smart cities in the EU[J]. European Parliament; Directorate general for internal policies, policy department economic and scientific policy A
22. Navarro-Arribas G, Torra V, Erola A, Castella-Roca J (2012) User k-anonymity for privacy preserving data mining of query logs. Inf Process Manag 48(3):476–487
23. Parreira JX, Dhungana D, Engelbrecht G (2015) The role of RDF stream processing in an smart city ICT infrastructure–the Aspern smart city use case. The semantic web: ESWC 2015 satellite events, pp 343–352
24. Pipino L, Lee YW, Wang RY (2012) Data quality assessment. Commun ACM 4:211–218
25. Qin H, Li H, Zhao X (2010) Development status of domestic and foreign smart city. Glob Presence 9:50–52
26. Rahm E, Do HH (2000) Data cleaning: problems and current approaches. IEEE Data Eng Bull 23(4):3–13
27. Redman TC (1996) Data quality for the information age. Artech House, Boston, MA
28. Samarati P, Sweeney L (1998) Generalizing data to provide anonymity when disclosing information. In: Proceedings of SIGMOD-SIGACT-SIGART symposium on the principles of database systems
29. Santos H, Pinheiro P, McGuinness DL (2015) Contextual data collection for smart cities. In: Proceedings of the 6th workshop on semantics for smarter cities
30. Scannapieco M, Catarci T (2002) Data quality under a computer science perspective. Arch Comput 2:1–15
31. Snigdha C, Tanveer AF, Hima PK, Mukesh KM, Venkata S (2015) Cleansing a database system to improve data quality. US Patent US9,104709 B2
32. Su K, Li J, Fu H (2011) Smart city and the applications. In: Electronics, communications and control (ICECC), pp 1028–1031
33. Sweeney L (2002) Achieving k-anonymity privacy protection using generalization and suppression. J Uncertain Fuzziness Knowl Based Syst 10(5):571–588
34. Thomsen C, Pedersen TB (2009) Pygrametl: a powerful programming framework for extract-transform-load programmers. In: Proceedings of DOLAP, pp 49–56
35. Thusoo A, Sarma JS, Jain N, Shao Z, Chakka P, Zhang N, Murthy R (2010) Hive-a petabyte scale data warehouse using Hadoop. In: Proceedings of ICDE, pp 996–1005

36. Wand Y, Wang RY (1996) Anchoring data quality dimensions in ontological foundations. Commun ACM 39(11):86–95
37. Wong RC, Li J, Fu AWC, Wang K (2007) K-anonymity: an enhanced k-anonymity model for privacy preserving data publishing. In: Proceedings of SIGKDD, pp 754–759
38. Zanella A, Bui N, Castellani A, Vangelista L, Zorzi M (2014) Internet of things for smart cities. Internet Things J 1(1):22–32
39. Zaveri A, Rula A, Maurino A, Pietrobon R, Lehmann J, Auer S (2016) Quality assessment for linked data: a survey. Semantic Web J 7(1):63–93

Xiufeng Liu is a researcher in Department of Management Engineering at DTU. He received his PhD in Computer Science from Aalborg University, Denmark, in 2012. Between 2013 and 2014, he worked as a postdoctoral researcher at Waterloo University, Canada. His research areas are smart meter data analytics, data warehousing, and big data.

Alfred Heller is an associate professor in Department of Civil Engineering at DTU, and the vice center leader of the CITIES research center. He received PhD in Civil engineering from Technical University of Denmark in 2000. He has a broad research interest, including building energy, district heating, solar heating, energy systems, smart energy cities and districts, data management, IT developments, amongst Web solutions.

Per Sieverts Nielsen received his PhD in Energy Systems Analysis from Technical University of Denmark in 1996. Per has been working in the energy area for more than 20 years, both in Denmark and in New Zealand. Since 2012, he has been employed as a senior researcher at DTU. He has expertise in a wide range of energy area, including energy system, energy modeling, renewable energy systems and CO2 mitigation strategies, and biomass energy systems.