# POLITECNICO DI TORINO

Facoltà di Ingegneria
**Corso di laurea in Ingegneria Informatica (Computer Engineering)**

Tesi di Laurea Magistrale

# Open Government Data Quality Assessment


Supervisor:
Prof. Marco Torchiano

Student:
Francesca Iuliano

December 2015

# Index

*Chapter 1*

# 1.Introduction

The term "Open Data" is recent and is strictly related to the spread of the Word Wide Web and the birth of the "social network", that is, a set of applications in which the users become contents producers. These applications can be labelled as Web 2.0 and they allow to create a network of applications in which the users can produce contents and share them. The idea of the Open Data is that data should be freely available to everyone to be used and republished as they wish and without the restrictions from copyright or other mechanism of controls. The objective of the Open Data movement is similar to those of other "Open" movements such as Open Software, Open Content and Open Access. Open source and Open Data share both the need to give free access to the information, the first one gives access to the source code while the second one provides free access to the public data. The open data has even an ethical objective because it promotes information exchange, knowledge, freedom of thought and benefits for the community. There are many kinds of Open Data that have potential uses and applications, such as: cultural, science, finance, statistics, weather, environment and transport.

The open data, for the access of the information and public assets is needed for the development of the Open Government within public administrations. The open public data, called Open Government Data (OGD) are published with the aim to improve the transparency and the participation of the citizens in the administration of their nation. Open Government data is the result of our time, with increasingly urgent imperatives of transparency and accountability. It is interesting to note that the players that are most affected by this crisis of confidence are also those who are most sensitive to Open Government data: politicians and public bodies, companies in the field of energy, environment, transport, banking. Transparency is perceived as a response to a period of mistrust, or distrust, towards institutions and their representatives**.** The Open Government allows an improvement of relationship among the government and citizens. By opening up data, citizens are enabled to be much more directly informed and involved in decision-making and government can help drive the creation of innovative business and services that deliver social and commercial value.

One of the first obstacles to spreading of the open data is that they may be restricted for privacy and since that data are managed by public company, they could be reluctant to give information about their information assets. Another problem of the Open Government Data is their quality. In many cases, the data are not published, or you can find them but which are incomplete, inaccurate or erroneous. The data with a low-quality are data with a low potential while those with a high level of quality are useful to citizens interested in testing how the government manages public money and public resources by increasing the transparency and reducing corruption and illegality.

The legislative decree – 14 marzo 2013 (14march2013), n.33 regards the obligations of publicity, transparency and dissemination of information by public authorities. In the decree is made explicit the function of public utility of the Open information, indicated as "widespread forms of control on the pursuit of official duties and the use of public resources". With regard to publication requirements, it indicates an obligation to create in the home page of its corporate website a special section, called “Amministrazione Trasparente” (Transparent administration) in which relevant information will be published. In the clause n.37 of Legislative Decree, reference is made to the disclosure obligations related to public contracts and in particular, it is expected that within January 31th of each year are published summary tables in an open standard format freely available allowing you to analyse and process the data for statistical purposes. In addition, administrations transmit this information in digital format to the Authority for the supervision of public contracts. The format chosen by the Authority, for the transmission of such data, is the XML. Transmission is done by communicating the URL of publication of these data.

Given the importance of data on public contracts, especially at this time in history when media attention is especially focused on the spending of public money, we have chosen to analyse such data to evaluate their quality and to examine the effectiveness of that model with respect to them. It is evaluated the capability of the model to capture the (possible lack of) quality in the data to allow, possibly, their improvement. After careful scouting of the data, we decided to evaluate the public contracts that Italian universities publish on its corporate website. In our research we have found that, in some cases, although the “Amministrazione Trasparente”(Transparent administration) section exists, data are not posted and updated, in some other cases the data are made available but not in XML format

while, most Italian universities publish the data in the correct format. We started from these universities to conduct a competition based on the quality of the data made available to the community.

The work is divided essentially into two parts: the ETL part and the Quality assessment part. The Extract, Transform and Load (ETL) refers to a process in database and especially in data warehousing that *extract* data from homogeneous or heterogeneous data sources, *transforms* them for storing them in proper format or structure for querying and analysis purposes and *loads* them into the final target that could be a database, datamart or data warehouse.

In our case, the first part involves extracting the data from the XML files, this represents the most important aspect of the ETL process since extracting data correctly sets the stage for the success of subsequent processes. In the data transformation stage a series of rules and functions are applied to the extracted data in order to prepare them for loading into the end target and for using them in the quality assessment part. Finally, the load stage is used to write the resulting data to a target database. Since the database had never been designed before, this phase includes the design of the database that will store the data about public contract and that will be used to conduct quality analysis.

The second part of the work consist on the evaluation of the data previously loaded into the database. Among the different models, a new one is defined for the evaluation of public contracts data. For assessing the quality of open data are considered some intrinsic aspects of the data, that is, aspects of quality that can be quantified for any type of data and therefore do not depend on the type of data analysed and some other aspects that are closely linked to the data domain. After the definition of the dimensions and the metrics to measure the quality of the attributes, we evaluate the quality of the datasets provided by the Italian Universities and comparing them according to the different results, we will get a ranking that will tell us which are the Italian Universities that publish data with higher quality.

*Chapter 2*

# 2. State of the art

## 2.1 What are Open Data?

The idea behind the open data is that certain data should be freely available and accessible and they can be used and redistributed by anyone.

The definition provided by the Open Knowledge Foundation is:

*"Open data is data that can be freely used, re-used and redistributed by anyone - subject only, at most, to the requirement to attribute and sharealike."*

The key features of openness are:

- *Availability and access*: the data must be available as a whole and at no more than a reasonable reproduction cost, preferably by downloading over the internet. The data must also be available in a convenient and modifiable form.
- *Reuse and redistribution*: the data must be provided under terms that permit reuse and redistribution including the intermixing with other datasets. The data must be machine-readable.
- *Universal participation*: everyone must be able to use, reuse and redistribute — there should be no discrimination against fields of endeavour or against persons or groups. For example, 'non-commercial' restrictions that would prevent 'commercial' use, or restrictions of use for certain purposes (e.g. only in education), are not allowed.

Open data responds to a set of technical, economic and legal criteria and they have to have a licence that says they are open data. Without a licence, the data can't be reused. The licence might also say:

- That people who use the data must credit whoever is publishing it (this is called *attribution*).

- That people who mix the data with other data have to also release the results as open data (this is the meaning of *sharealike*).

There are many kinds of open data that have potential uses and applications:

- *Cultural***:** Data about cultural works and artefacts — for example titles and authors — and generally collected and held by galleries, libraries, archives and museums.
- *Science*: Data that is produced as part of scientific research from astronomy to zoology.
- *Finance***:** Data such as government accounts (expenditure and revenue) and information on financial markets (stocks, shares, bonds etc).
- *Statistics***:** Data produced by statistical offices such as the census and key socioeconomic indicators.
- *Weather***:** The many types of information used to understand and predict the weather and climate.
- *Environment***:** Information related to the natural environment such presence and level of pollutants, the quality of rivers and seas.
- *Transport***:** Data such as timetables, routes, on-time statistics.

It is important to highlight that the creation of value depends on the ability to share data, to make it available to third parties. Open data, indeed, enables third parties to create innovative products and services using datasets such as transportation data, or data about medical treatments and their outcomes, that are generated in the course of providing public services or conducting research. Some of the key motivations for open data initiatives are to promote transparency of decision-making, create accountability for elected and spur greater citizen engagement. This data can not only be used to help increase the productivity of existing companies and institutions, it also can spur the creation of entrepreneurial businesses and improve the welfare of individual consumers and citizens.

## 2.2 What are Open Government Data?

The philosophy of *Open Government* is based on the principle that all activities of the government and public administration must be open and available to facilitate effective action and ensure control over the management of public affairs. It redefines

the relationship between citizens and government, the citizen is no longer a simple consumer of services but he participates in the government's choices.

The open Government is based on three main principles:

- *Transparency*: It promotes accountability by providing to citizens the information on the activities of the administration. An Open Administration is then more controlled and more reliable in the meantime. For this reason, governments that move in the direction of Open Government should take all measures to ensure that the information they have are made available, reusable and open.
- *Participation*: The participation of citizens in the choices of the government increases the effectiveness of administrative action and improves the quality of decisions. Citizens must be involved in decision-making process to contribute actively through the use of communication technologies currently available.
- *Collaboration*: The collaboration promotes the direct involvement of citizens in the activities of government. This collaboration gives the possibility to citizens to monitor the quality of public service in all its phases of execution.

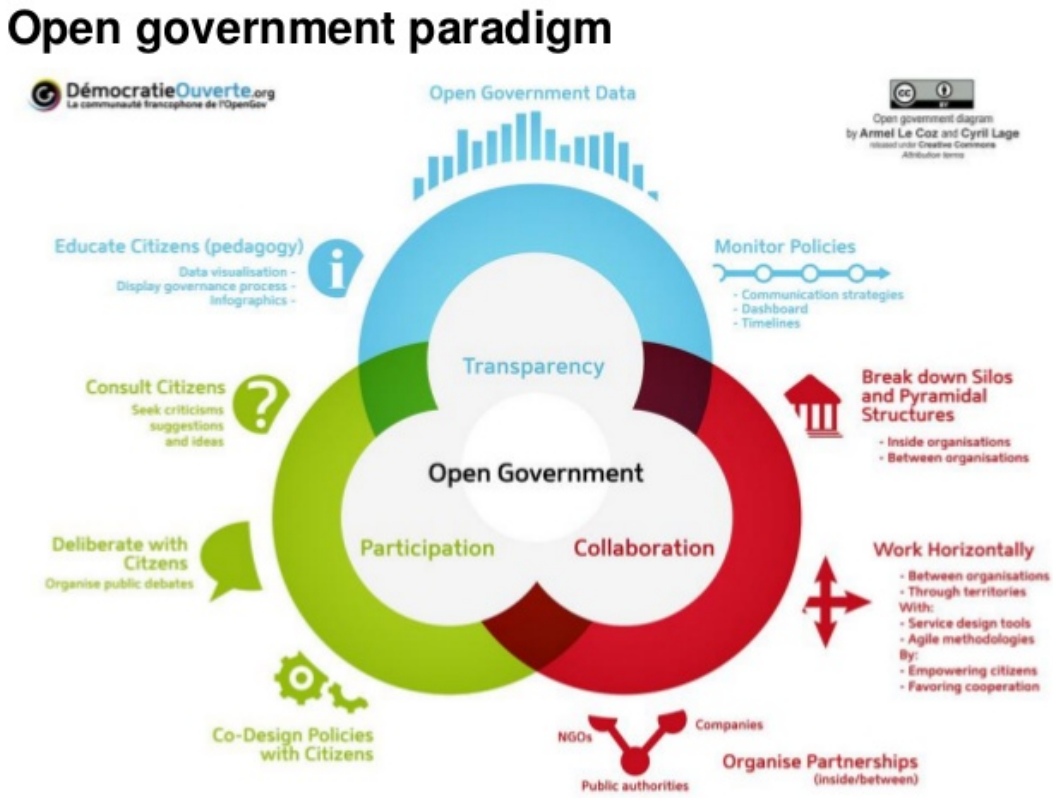



**Figure 1:** Open Government Paradigm

To implement the principles of transparency, participation and collaboration of the Open Government is necessary to provide to the citizens the tools to make decisions or, otherwise, to evaluate the decisions taken by the administration. These tools are the Open Government Data, information held by the government and which are essential for process management.

More precisely, Open Government Data means:

- Data produced or commissioned by government or government controlled entities;
- Data which is Open as defined by the Open Definition.

Government Data shall be considered open if it is made public in a way that complies with the principles below:

- *Complete*: All public data is made available. Public data is data that is not subject to valid privacy security or privilege limitations.
- *Primary*: Data is as collected at the source, with the highest possible level of granularity, not in aggregate or modified forms.
- *Timely*: Data is made available as quickly as necessary to preserve the value of the data.

- *Accessible***:** Data is available to the widest range of users for the widest range of purposes.
- *Machine processable***:** Data is reasonably structured to allow automated processing.
- *Non – discriminatory***:** Data is available to anyone, with no requirement of registration.
- *Non – proprietary***:** Data is available in a format over which no entity has exclusive control.
- *Licence - free***:** Data is not subject to any copyright, patent, trademark or trade secret regulation. Reasonable privacy, security and privilege restrictions may be allowed.

The principles mentioned above make us realize that Open Government Data as well as create great benefits, have also many obstacles that limit their spread. There are administrative and legal barriers that prevent the publication of certain types of data. Because government information is a mix of public records, personal information, copyrighted work and other non-open data, it is important to be clear about what data is available and what licensing, terms of service, and legal restrictions apply. Data for which no restriction apply should be marked clearly as being in the public domain. Furthermore, the data that are not subject to privacy must be continually updated in order to have value. The awareness of the economic and social potential of open data is relatively low. The authorities are not always willing to modify the processes in use and do not perceive the importance to their role as producers of data. Published data must be of the highest quality possible, that is, they must meet high standard of completeness, accuracy and consistency. To get data with high quality it is necessary to invest in people with expertise and specialized skills and this may increase the production costs. Another problem is the lack of knowledge of the citizens about the published data that leads to a partial and incomplete use of the published information.

## 2.3 A brief history of Open Government Data

The term open data appeared for the first time in 1995, in a document from an American scientific agency. It dealt about the disclosure of geophysical and environmental data. They promote a complete and open exchange of scientific information between different countries, a prerequisite for the analysis and understanding of these global phenomena. The idea of common good applied to

knowledge had already been theorized well before the invention of the Internet. Robert King Merton was one of the fathers of sociology of science and the theory that bears his name shows the benefits of open scientific data. As early as 1942, Merton explained the importance that the results of research should be freely accessible to all. Each researcher must contribute to the "common pot" and give up intellectual property rights to allow knowledge to move forward. Information technologies have also given a new breath to this philosophy of commons. Long before being a technical object or political movement, open data was rooted in the praxis of the scientific community. Researchers were the first who perceived the benefit of openness and of sharing of data.

In December 2007, thirty thinkers and activists of the Internet held a meeting in Sebastopol, north of San Francisco. Their aim was to define the concept of open public data and have it adopted by the US presidential candidates. Among them, were two well-known figures: Tim O'Reilly and Lawrence Lessig. The first one is an American author which defined and popularized expressions such as open source and Web 2.0. Lawrence Lessig, Professor of Law at Stanford University (California), is the founder of Creative Commons licenses, based on the idea of copyleft and free dissemination of knowledge. Some activists and entrepreneurs who already used public data were attending the Sebastopol meeting too and together, they created the principles that allow us today to define and evaluate open public data. Furthermore, Tim O'Reilly's contribution on Open Government shed a new light on the relation between the open source movement and the emerging principles of Open Data. In his own words:

*"we must apply the principles of open source and its working methods to public affairs".*

In 2007, it sounded like a dream. But the result has exceeded by far their expectations. A little over a year later, President Barack Obama took office in the White House and signed three 'presidential memoranda'. Two of them concern Open Government, of which Open Data is one of the pillars. These presidential memos explicitly set the culture of open source at the heart of public action by claiming its founding principles: transparency, participation and collaboration. President Barack Obama declared:

*"openness will strengthen our democracy and promote efficiency and effectiveness in government."* (Obama, Memorandum for the heads of executive departments and agencies, 2009).

America has been a model for other countries, such as Canada and Australia, which in a few months have created their own data stores.

The Openness history in Italy began in 2002 with the draft law on the distribution of free software. In 2005 the Open Source Software Committee came and produced a survey of open source software by which have been defined several decrees and regulations. In these decrees it is perceived not only the software openness but also the data openness importance. In 2005, the Italian legislation, already asked to the public administrations to adopt open formats but, perhaps, the Italian Government was not ready and it has not grasped the importance of the openness. Only a few years later it was launched the process towards opening data. The Piedmont Region has created, in 2010, the first regional website for open data. That website remains the most successful experience on the subject and open data. In 2011 the Region of Emilia Romagna has followed the example of the Piedmont and has created their own Open Data website. In the following years the other regions have begun the process of adapting, creating their data stores.

## 2.4 State of the art of the Open Government Data

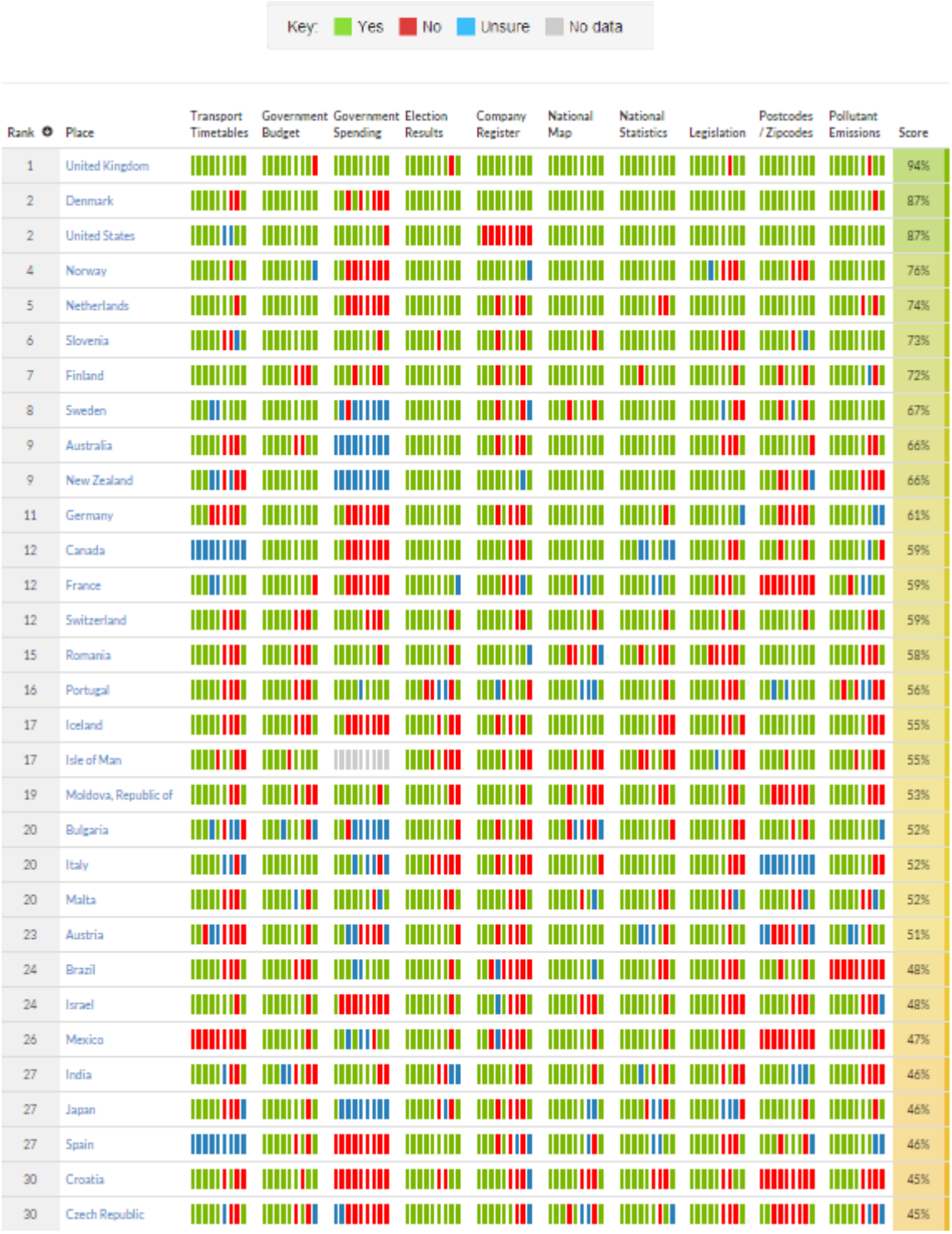
An increasing number of governments have committed to open up data but it is important to understand how much of key information is actually being released and which are the most advanced countries.

The Open Data Index is an annual expert peer-reviewed snapshot of the country-level Open Data Census, and has been developed to help answer previous questions by collecting and presenting information on the state of open data around the world. In the following we will use the Open Data Index to compare different countries taking as reference the 2013 and the 2014.

With the Open data Index it is evaluated if the data:

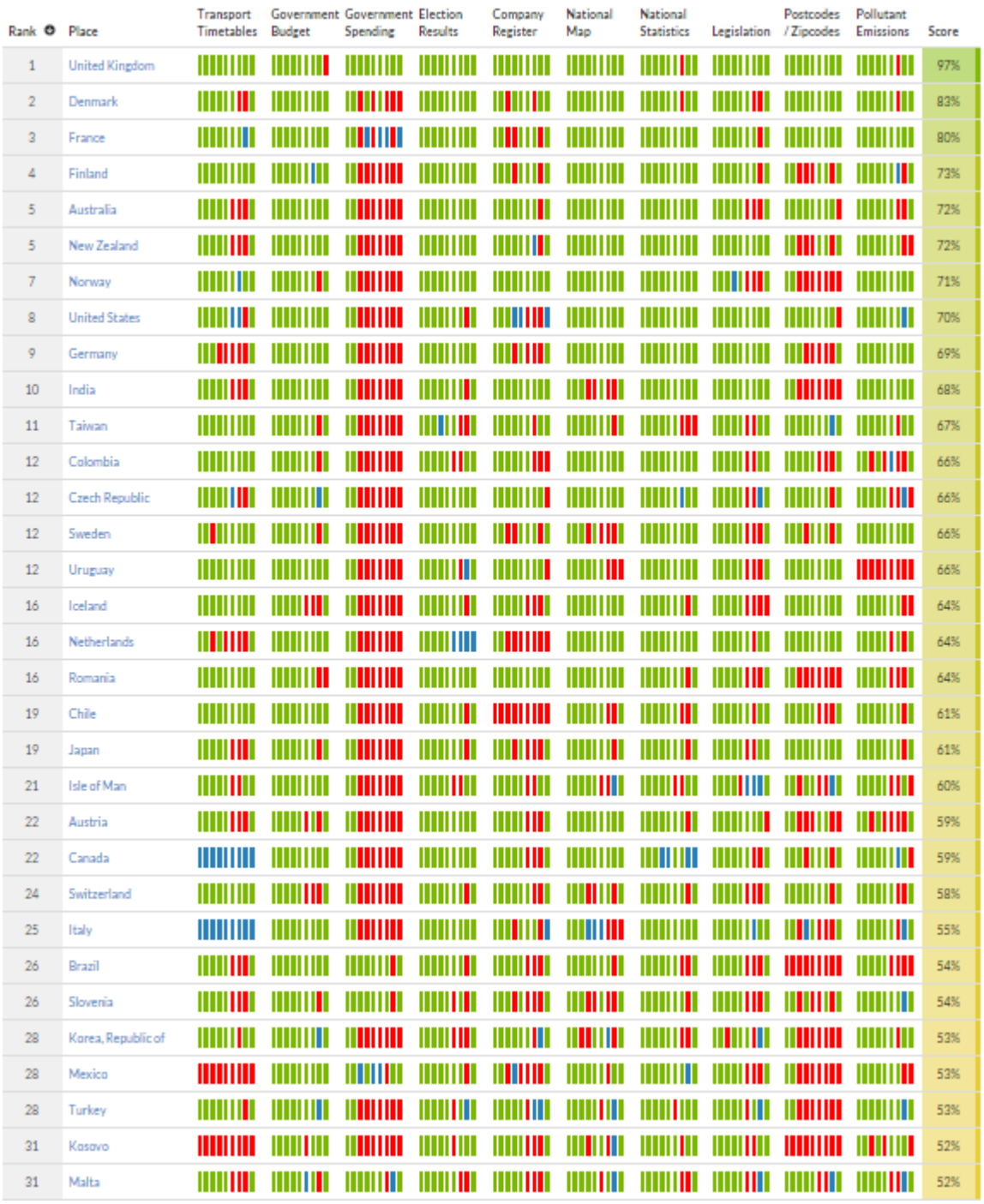
- Exists;
- Is digital;
- Is public;

- Is free;
- Is online;
- Is machine-readable;
- Is available in bulk;
- Is openly licensed;
- Is up to date.




| Rank | Place | Transport Timetables | Government Budget | Government Spending | Election Results | Company Register | National Map | National Statistics | Legislation | Postcodes / Zipcodes | Pollutant Emissions | Score |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | United Kingdom | | | | | | | | | | | 94% |
| 2 | Denmark | | | | | | | | | | | 87% |
| 2 | United States | | | | | | | | | | | 87% |
| 4 | Norway | | | | | | | | | | | 76% |
| 5 | Netherlands | | | | | | | | | | | 74% |
| 6 | Slovenia | | | | | | | | | | | 73% |
| 7 | Finland | | | | | | | | | | | 72% |
| 8 | Sweden | | | | | | | | | | | 67% |
| 9 | Australia | | | | | | | | | | | 66% |
| 9 | New Zealand | | | | | | | | | | | 66% |
| 11 | Germany | | | | | | | | | | | 61% |
| 12 | Canada | | | | | | | | | | | 59% |
| 12 | France | | | | | | | | | | | 59% |
| 12 | Switzerland | | | | | | | | | | | 59% |
| 15 | Romania | | | | | | | | | | | 58% |
| 16 | Portugal | | | | | | | | | | | 56% |
| 17 | Iceland | | | | | | | | | | | 55% |
| 17 | Isle of Man | | | | | | | | | | | 55% |
| 19 | Moldova, Republic of | | | | | | | | | | | 53% |
| 20 | Bulgaria | | | | | | | | | | | 52% |
| 20 | Italy | | | | | | | | | | | 52% |
| 20 | Malta | | | | | | | | | | | 52% |
| 23 | Austria | | | | | | | | | | | 51% |
| 24 | Brazil | | | | | | | | | | | 48% |
| 24 | Israel | | | | | | | | | | | 48% |
| 26 | Mexico | | | | | | | | | | | 47% |
| 27 | India | | | | | | | | | | | 46% |
| 27 | Japan | | | | | | | | | | | 46% |
| 27 | Spain | | | | | | | | | | | 46% |
| 30 | Croatia | | | | | | | | | | | 45% |
| 30 | Czech Republic | | | | | | | | | | | 45% |

**Figure 2:** Open Data Index for the 2013 (provided by OKFN)



| Rank | Place | Transport Timetables | Government Budget | Government Spending | Election Results | Company Register | National Map | National Statistics | Legislation | Postcodes / Zipcodes | Pollutant Emissions | Score |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | United Kingdom | | | | | | | | | | | 97% |
| 2 | Denmark | | | | | | | | | | | 83% |
| 3 | France | | | | | | | | | | | 80% |
| 4 | Finland | | | | | | | | | | | 73% |
| 5 | Australia | | | | | | | | | | | 72% |
| 5 | New Zealand | | | | | | | | | | | 72% |
| 7 | Norway | | | | | | | | | | | 71% |
| 8 | United States | | | | | | | | | | | 70% |
| 9 | Germany | | | | | | | | | | | 69% |
| 10 | India | | | | | | | | | | | 68% |
| 11 | Taiwan | | | | | | | | | | | 67% |
| 12 | Colombia | | | | | | | | | | | 66% |
| 12 | Czech Republic | | | | | | | | | | | 66% |
| 12 | Sweden | | | | | | | | | | | 66% |
| 12 | Uruguay | | | | | | | | | | | 66% |
| 16 | Iceland | | | | | | | | | | | 64% |
| 16 | Netherlands | | | | | | | | | | | 64% |
| 16 | Romania | | | | | | | | | | | 64% |
| 19 | Chile | | | | | | | | | | | 61% |
| 19 | Japan | | | | | | | | | | | 61% |
| 21 | Isle of Man | | | | | | | | | | | 60% |
| 22 | Austria | | | | | | | | | | | 59% |
| 22 | Canada | | | | | | | | | | | 59% |
| 24 | Switzerland | | | | | | | | | | | 58% |
| 25 | Italy | | | | | | | | | | | 55% |
| 26 | Brazil | | | | | | | | | | | 54% |
| 26 | Slovenia | | | | | | | | | | | 54% |
| 28 | Korea, Republic of | | | | | | | | | | | 53% |
| 28 | Mexico | | | | | | | | | | | 53% |
| 28 | Turkey | | | | | | | | | | | 53% |
| 31 | Kosovo | | | | | | | | | | | 52% |
| 31 | Malta | | | | | | | | | | | 52% |

**Figure 3:** Open Data Index for 2014 (provided by OKFN)

As we can see from Figure 2 and Figure 3, the best state regarding the data openness is the United Kingdom. Compared to 2013, The United States of America are located in seventh place overtaken by Denmark, France, Finland, Australia, New Zealand and Norway. Looking at Italy, we can say that there has been a slight improvement compared to 2013, but these improvements are still slow compared to other countries.

| Rank | Dataset | Breakdown | Location (URL) | Format | Info | Prev. (2013) | Score |
|---|---|---|---|---|---|---|---|
| 1 | Government Budget | | http://www.rgs.mef.gov.it/VERS... | CSV, Excel | | #1 100% | 100% |
| 1 | National Statistics | | http://dati.istat.it/ | XLS, RDF, ... | n/a | #1 100% | 100% |
| 1 | Election Results | | http://elezioni.interno.it/ope... | CSV | | #54 30% | 100% |
| 6 | Legislation | | http://www.normattiva.it/ | XML | | #24 50% | 90% |
| 16 | Government Spending | | n/a | n/a | | #18 20% | 10% |
| 23 | Pollutant Emissions | | http://www.brace.sinanet.apat.... | XLS, CSV | | #13 60% | 60% |
| 26 | Company Register | | http://www.registroimprese.it/... | n/a | | #39 30% | 45% |
| 36 | Postcodes / Zipcodes | | http://www.poste.it/online/cer... | n/a | | #45 0% | 25% |
| 65 | Transport Timetables | | n/a | n/a | | #34 35% | 0% |
| 67 | National Map | | http://www.pcn.minambiente.it/... | wfs, wms | | #13 90% | 15% |

**Figure 4:** Open data Index Italy 2014(provided by 2014)

As we can see from the figure 3, no country reaches the full points. The maximum reached is seven complete dataset on a total of ten on which the analysis is carried out. The majority of the dataset are not openly licensed and they are not machine – readable. The OKFN states that, for the 2014, only the 15% of the dataset entries are open as defined by the Open Definition. The most interesting thing is that for the first twenty country of the rank, except United Kingdom, the data about governing spending are those less published and as we told before, this reduce the transparency of an Open Government.

## 2.5 The Transparency Decree

April 5, 2013 was published the legislative decree 14 marzo 2013, n.33 (14 march 2013) on the reorganization of the previous law concerning the obligations of publicity, transparency and dissemination of information by public authorities for a total accessibility of information about the activities of public administrations. Here we look at the provisions that most closely relate to contracts for public works, services and supplies. Article 37 of the Decree recall, with some adjustments, Article 1 paragraph 32 of Law 190/2012. The law 190/2012 is about dispositions for the prevention and suppression of corruption and illegality in public administration and states that contracting authorities are, in any case obliged to publish on their institutional website the proposer, the object of the contract, the list of participants, the contractor, the award amount, timing of completion of service or supply and the amount paid. Within January 31th each year, the information relating to the previous year, are published in summary tables made publicly available for download in a digital open standard format that allows to analyse and process those data. The administrations shall send, in digital format, such information to the Supervisor of

public contracts which will post them on its website, in a section that can be consulted by all citizens, categorized according to the type of contracting authority and by region. The authority identifies and specifies the relevant information and the method of transmission. Within April 30th each year, the authority for the supervision of public contracts shall send to the Court of Auditors the list of administrations that have failed to transmit and publish, in whole or in part, the information in digital open standard format. In this regard, the authorities provided, by resolution 26 of 22 May 2013, the specific set of data required and its format, making it clear that the data transmission means absolved by communicating the URL of publication of these data, in XML format.

Article 9 of the Transparency Decree brings order to the previous law on the access to information published on the institutional websites of each public administration. For the full accessibility of the published information on the home page of every corporate website, is located a section called 'Amministrazione Trasparente'(Transparent Administration) in which are contained the data, information and documents. Administrations may not have filters and other technical means to prevent the web search engines to search within that section. When the duration of the exposure of such data expires, the data is moved in the archive and this is reported within the section ‘Amministrazione Trasparente’. The 'Amministrazione Trasparente' (Transparent Administration') section must be organized into sub sections within which there must be the documents, information and data as specified by the decree. The data about public contracts are published in the ‘Bandi di gara e contratti' as specified in Article 1 paragraph 32 of Law 190/2012. Since the entry into force of this decree, many governments have made efforts to fulfil the obligations of publication. For the purposes of being able to monitor the transparency obligations imposed by the decree, the government has made available a tool for analysis and monitoring of websites called ‘Bussola della Trasparenza’ (Transparency compass).

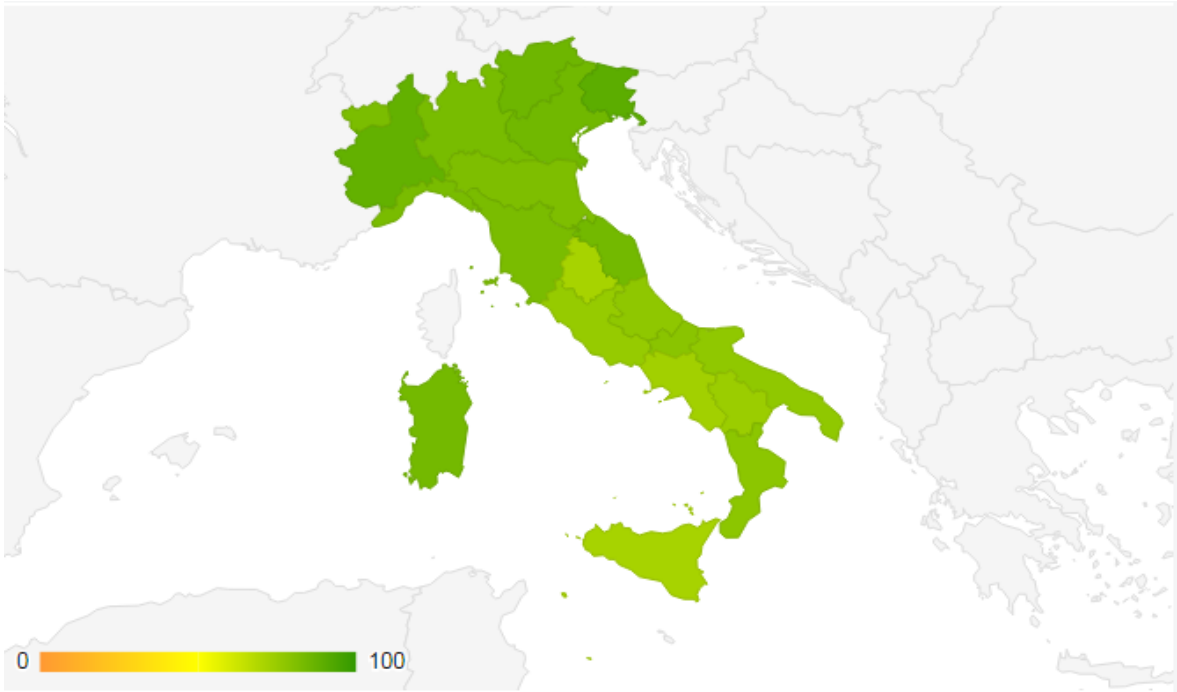



**Figure 5:**Colours of transparency

Figure 5 shows the percentage of public administrations, divided by regions, which respect the structure of the sub-section 'Bandi di gara e contratti' imposed by transparency decree. Given that this tool only checks that the structure of websites respects what is specified in the decree, in many cases is not effective. Many governments have structured its website to meet the requirements of the decree but within the section there aren't data. It would therefore be more useful to find out what data and how they are published by public administrations.

*Chapter 3*

# 3. Extraction, transformation and loading

The extraction, transformation and loading is a three-stage process in database usage and data warehousing. First, the extract function reads data from a specified source and extract a desired dataset of data. Next, the transform function works with the acquired data, using rules, lookup tables or creating combination with other data, to convert it to desired state. Finally, the load function is used to write the resulting data, either the entire dataset or just the changes, to a target database which may or may not previously exists. The data loaded using the ETL can be then used for analysis purposes. Even if the three phases are represented as being executed sequentially, usually they execute in parallel. Since the data extraction takes time, while the data is being pulled another transformation process executes processing the already received data and prepares the data for loading and, as soon as there is some data ready to be loaded into the target, the data loading are stored without waiting the completion of the previous phases.

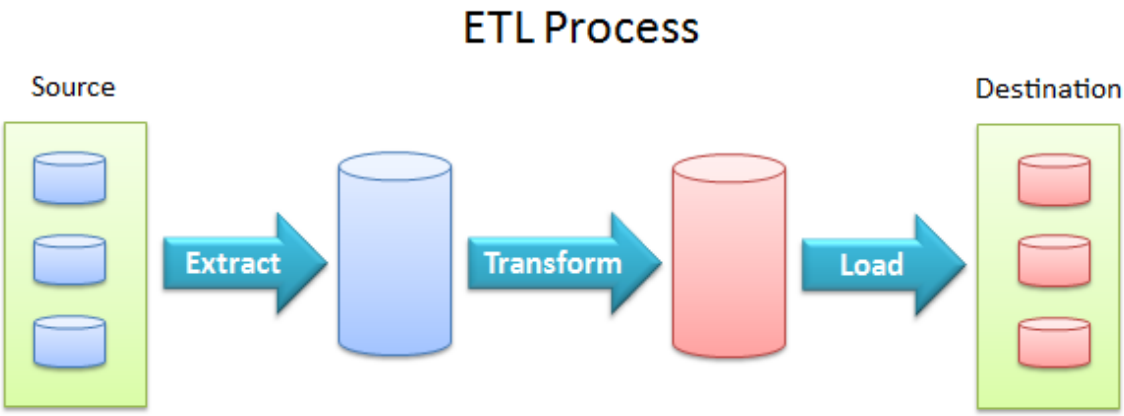



**Figure 6:**Extraction Transformation and Loading Process

To lead our race, where it is evaluated the quality of data relating to public contracts made available by the Italian Universities we use the ETL process.

The technologies chosen to develop our work are:

- *Java Language*: it is used at various stages, the access and extraction of data, during the transformation of the data in a format that enables the analysis and also in the analysis part.
- *Mysql*: to store the information previously collected, and manipulated. The database is used as a starting point to perform the analysis.

*Java* was first developed by James Gosling at Sun Microsystems which has been acquired by Oracle Corporation. Released in the 1995, is a general-purpose computer programming language that is:

- Concurrent;
- Class-Based;
- Object-Oriented.

It was specifically designed to have as few implementation dependencies as possible. The language derives much of its syntax from C and C++ but it has fewer low-level facilities than either of them. As of 2015, *Java* is one of the most popular programming languages in use.

*MySQL* is an open source and free database and is one of the most known and widespread technologies in the IT world. *MySQL* was founded in 1996 by a Swedish company based on a pre-existing relational DBMS called *mSQL*. The project is distributed as open source to promote growth. Since 1996, *MySQL* has had great success and the ways of this success are:

- High efficiency despite the big size of data;
- Integration of all the features that a DBMS can offer such as index, trigger and stored procedure;
- High integration capabilities with the main programming languages and development environments.

## 3.1 Extraction stage

The first step is to look for source files. The law 190/2012 requires each contracting authority to publish, within January 31th each year, summary tables about the

previous year and send the same to the Supervisor of public contracts, leaving the Authority to disclose the method of transmission and the data format required. The A.N.A.C., the Authority for the supervision of public contracts, states that the publication of the dataset by the contracting authority, must take place on its corporate website. If the size of the single dataset exceed 5 MB, an index document containing the URL of the individual datasets must be provided. The first activity to perform during the extraction step is, therefore, to search datasets in each institutional websites of each analysed University. Once the XML file has been collected, we can proceed with the extraction of the data from such files. To extract the data we need to understand how they are organized within a single file thus, we need to know the structure of the file and which type of data are stored in the files. For this purpose, the Authority provides the XML Schema, then all the published file must comply with, for each field, the data type and the maximum length specified in that schema.

### 3.1.1 XML schema

When sending data from sender to receiver, it is essential that both parts have the same expectations about the content. With the XML schemas the sender can describe the data in a way that the receiver will understand. XML Schema Definition (XSD) language is the current standard schema language for all XML documents and data. The XML Schema Definition allows you to define the structure and data types for XML documents.

An XML Schema defines:

- Elements and attributes that can appear in a document;
- Which elements are child elements;
- The order of child elements;
- Whether an element is empty or can include text;
- Data types for elements and attributes;
- Default and fixed values for elements and attributes.

One of the greatest strength of XML Schemas is the support for data types, with them it is easier:

- To describe allowable document content;
- To evaluate the correctness of data;

- To work with data from a database;
- To define data facets, that is, restrictions on the data;
- To define data patterns (data format);
- To convert data between different data types.

The XSD provided by the A.N.A.C shows that the structure of the file must contain:

- A section with the metadata of the dataset.
- A data section containing the list of lots, more precisely:
    a. Each record corresponds to a lot. The record is tree-structured, that is, exist for each lot a number of child records that can report information with variable cardinality.
    b. Each lot consists of several elements.
    c. Each file can contain multiple lots.

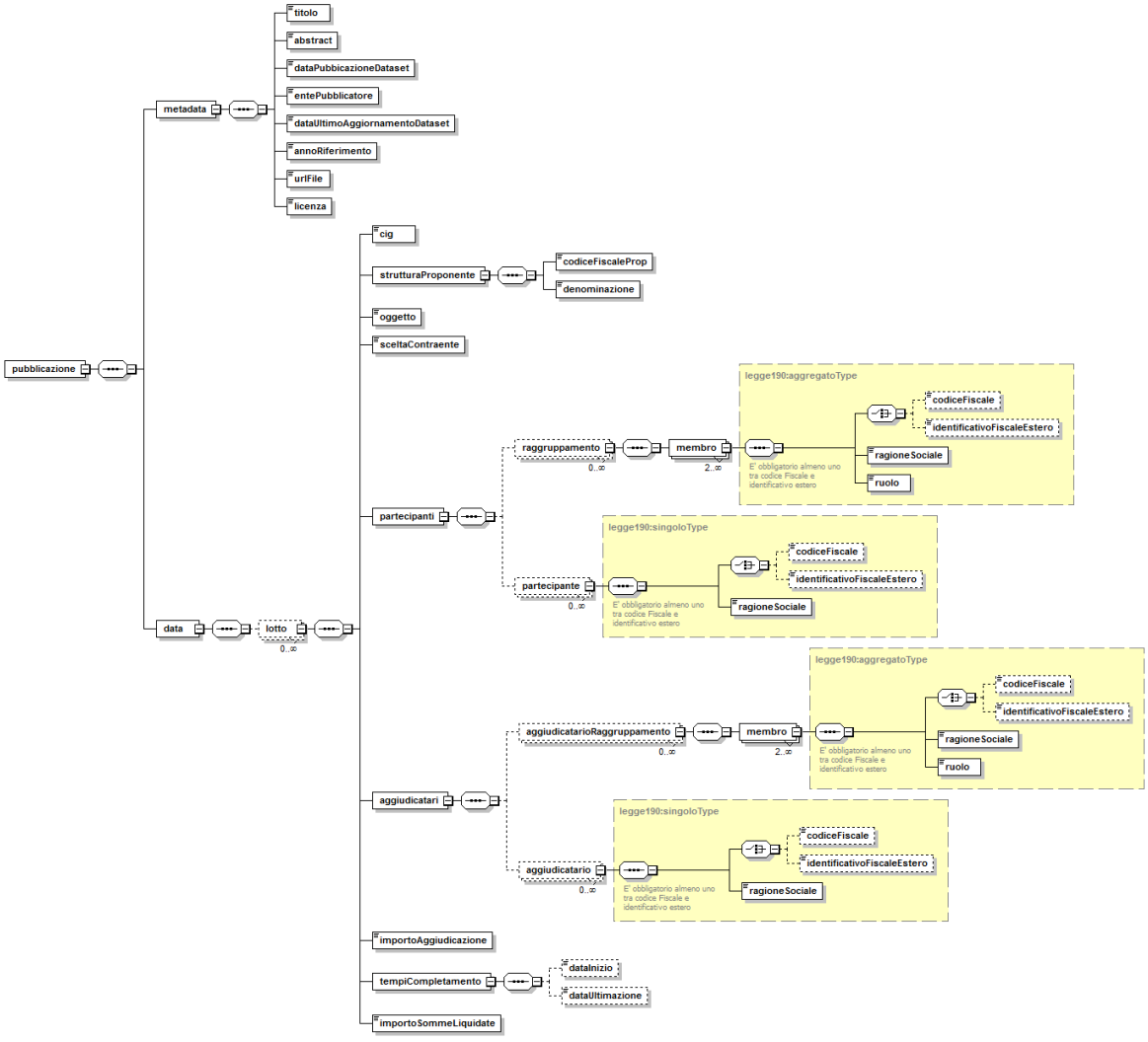



**Figure 7:** XSD schema of the public contract dataset

The Figure 7 shows how it should be structured an xml file. In the following we will give a short description of the meaning of each element of both metadata and data sections.

### Start of <Metadata>

| XML Tag | Description |
|---|---|
| Titolo | Title of the dataset. |
| Abstract | Brief description. |
| dataPubblicazioneDataset | Date in which the dataset has been made public. |
| entePubblicatore | Contracting authority which publishes the dataset. |
| dataUltimoAggiornamentoDataset | Date of last update of the dataset. |
| annoRiferimento | The year data is related. |
| urlFile | Url of the dataset. |
| Licenza | License. |

### End of <Metadata>

### Start of <data>/<lotto>

| XML Tag | Description |
|---|---|
| cig | Identification code of the public contract given by the Authority. |

### Start of <data>/<lotto>/<strutturaProponente>

| XML Tag | Description |
|---|---|
| codiceFiscaleProp | Tax code of the contracting Authority. |
| denominazione | Name of the contracting Authority. |

### End of <data>/<lotto>/<strutturaProponente>

| XML Tag | Description |
|---|---|
| oggetto | Subject of the lot identified by the cig. |
| sceltaContraente | Procedure of contractor selection. |

### Start of <data>/<lotto>/<partecipanti>
### <raggruppamento>/<membro>

| XML Tag | Description |
|---|---|
| codiceFiscale | Tax code of the company which participates in the contractor selection. |
| identificativoFiscaleEstero | Foreign tax code of the company which participates in the contractor selection. |
| ragioneSociale | Name code of the company which participates in the contractor selection. |
| ruolo | Role when participating with others. |

### <partecipante>

| XML Tag | Description |
|---|---|
| codiceFiscale | Tax code of the company which participates in the contractor selection. |

| | |
|---|---|
| identificativoFiscaleEstero | Foreign tax code of the company which participates in the contractor selection. |
| ragioneSociale | Name of the company which participates in the contractor selection. |

**End of <data>/<lotto>/<partecipanti>**

**Star of <data>/<lotto>/<aggiudicatari>**

**<aggiudicatarioRaggruppamento>/<membro>**

| XML Tag | Description |
|---|---|
| codiceFiscale | Tax code of the company that wins the contractor selection. |
| identificativoFiscaleEstero | Foreign tax code of the company that wins the contractor selection. |
| ragioneSociale | Name of the company that wins the contractor selection. |
| ruolo | Role when participating with others. |

**<aggiudicatario>**

| XML Tag | Description |
|---|---|
| codiceFiscale | Tax code of the company that wins the contractor selection. |
| identificativoFiscaleEstero | Foreign tax code of the company that wins the contractor selection. |
| ragioneSociale | Name of the company that wins the contractor selection. |

**End of <data>/<lotto>/<aggudicatari>**

| XML Tag | Description |
|---|---|
| importoAggiudicazione | Award amount. |

**Start of <data>/<lotto>/<tempiCompletamento>**

| XML Tag | Description |
|---|---|
| dataInizio | Actual date of the beginning of work, services or supplies. |
| dataUltimazione | Completion date of work, services or supplies. |

**End of <data>/<lotto>/<tempiCompletamento>**

| XML Tag | Description |
|---|---|
| importoSommeLiquidate | Amount payed. |

There are two ways to verify whether XML files are coded correctly:

- *Well-Formedness*: the XML file must be syntactically correct.
- *Validity*: if the XML file has an associated XML Schema, the element must appear in the defined structure and the content of the individual elements must conform to the declared data types specified in the schema.

To generate valid XML files, it is important to note that some elements, such as all the elements in the metadata sections or the cig and the codiceFiscale elements and

many other are required and cannot be omitted while, other elements, such as dataInizio and dataCompletamento are optional and can be omitted.

```
<?xml version="1.0" encoding="UTF-8"?>
<legge190:pubblicazione xsi:schemaLocation="legge190_1_0 datasetAppaltiL190.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:legge190="legge190_1_0">

<metadata>
<titolo> Pubblicazione 1 legge 190</titolo>
<abstract> Pubblicazione 1 legge 190 anno 1 rif. 2010</abstract>
<dataPubbicazioneDataset>2012-08-13</dataPubbicazioneDataset>
<entePubblicatore>AVCP</entePubblicatore>
<dataUltimoAggiornamentoDataset>2012-09-15</dataUltimoAggiornamentoDataset>
<annoRiferimento>2012</annoRiferimento>
<urlFile> http://www.pubblicazione.it/dataset1.xml </urlFile>
<licenza> IODL</licenza>
</metadata>

<data>
<lotto>
<cig>4939483E4E</cig>
<strutturaProponente>
<codiceFiscaleProp>97163520584</codiceFiscaleProp>
<denominazione>Autorità per la Vigilanza sui Contratti Pubblici di Lavori, Servizi e
Forniture</denominazione>
</strutturaProponente>
<oggetto>Gara a procedura aperta per l'affidamento della Fornitura di infrastrutture informatiche per il
programma AVCPass</oggetto>
<sceltaContraente>17-AFFIDAMENTO DIRETTO EX ART. 5 DELLA LEGGE
N.381/91</sceltaContraente>
<partecipanti>
<raggruppamento>
<membro>
<codiceFiscale>00000000001</codiceFiscale>
<ragioneSociale>Azienda 1</ragioneSociale>
<ruolo>04-CAPOGRUPPO</ruolo>
</membro>
<membro>
<codiceFiscale>00000000002</codiceFiscale>
<ragioneSociale>Azienda 2</ragioneSociale>
<ruolo>03-ASSOCIATA</ruolo>
</membro>
</raggruppamento>
<partecipante>
<codiceFiscale>00000000003</codiceFiscale>
<ragioneSociale>Azienda Individuale 1</ragioneSociale>
</partecipante>
</partecipanti>
<aggiudicatari>
<aggiudicatarioRaggruppamento>
<membro>
<codiceFiscale>00000000001</codiceFiscale>
<ragioneSociale>Azienda 1</ragioneSociale>
<ruolo>04-CAPOGRUPPO</ruolo>
</membro>
<membro>
<codiceFiscale>00000000002</codiceFiscale>
<ragioneSociale>Azienda 2</ragioneSociale>
<ruolo>03-ASSOCIATA</ruolo>
</membro>
</aggiudicatarioRaggruppamento>
</aggiudicatari>
<importoAggiudicazione>1000.00</importoAggiudicazione>
<tempiCompletamento>
<dataInizio>2012-08-13</dataInizio>
<dataUltimazione>2012-08-13</dataUltimazione>
</tempiCompletamento>
```

```
<importoSommeLiquidate>1000.00</importoSommeLiquidate>
</lotto>
</data>
</legge190:pubblicazione>
```

The above example shows a valid XML file which contains only one lot and all the elements specified by the XSD.

Once learned about the structure of the file we can proceed with the second phase of the extraction step that is the access and extraction of the information from the XML files.

### 3.1.2 Java Architecture for XML Binding

The Extensible Markup Language (XML) and Java technologies are natural partners in helping developers exchange data and programs across the Internet. That's because XML has emerged as the standard for exchanging data across disparate systems, and Java technology provides a platform for building portable applications. Given the widespread use of XML it is important to understand how it is possible to access and use XML files through the Java programming language. One way to do this, perhaps the most typical, is through a parser that conforms to the Document Object Model (DOM). This parser is provided by Java API for XML Processing (JAXP). Java developers can invoke a DOM parser in an application through the JAXP API to parse an XML document, that is, scan the document and logically break it up into discrete pieces. The parsed content is then made available to the application. Using this approach, the parser creates a tree of objects that represents the content and organization of data in the document. In this case, the tree exists in memory. The application can then navigate through the tree to access the data it needs, and if appropriate, manipulate it. Now developers have another Java API at their disposal that can make it easier to access XML: Java Architecture for XML Binding (JAXB). JAXB allows Java developers to access and process XML data without having to know XML or XML processing. In order to access the information and be able to manipulate them, only two steps are required:

- *Bind the schema*: for an XML document.
- *Unmarshal the document* into Java Contents Objects. The Java Contents Objects represent content and organization of the XML document, and are directly available to the program.

After unmarshalling, the program can access and display the data in the XML document simply by accessing the data in the Java Content Objects an then displaying them.

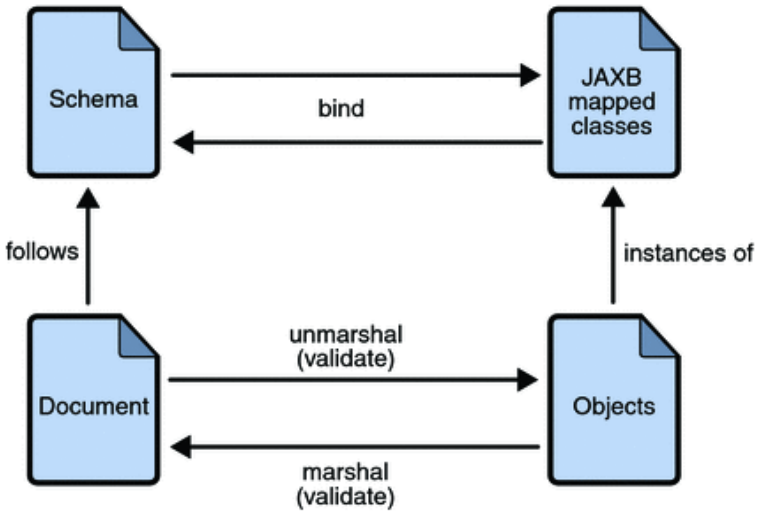



**Figure 8:** JAXB Architecture (provided by https://docs.oracle.com/javase/tutorial/jaxb/intro/arch.html)

### 3.1.2.1 Bind the schema

JAXB simplifies access to an XML document from a Java program by presenting the XML document to the program in a Java format. The first step in this process is to bind the Schema for the XML document into a set of Java classes that represents the schema. Binding a schema means generating a set of Java classes that represents the schema. JAXB implementations provide a tool called a binding compiler to bind a schema. The JAXB Reference Implementation provides a binding compiler that can be invoked through scripts. The command used to run the script that bind the schema is:

```
xjc [-Options] Schema
```

### 3.1.2.2 Unmarshal the Document

Unmarshalling an XML document means creating a tree of content objects that represents the content and organization of the document. The content objects are instances of the classes produced by the binding compiler. In addition to providing a binding compiler, a JAXB implementation must provide runtime APIs for JAXB-related operations such as marshalling.

To unmarshal an XML document:

- Create a *JAXBContext* object. This object provides the entry point to JAXB API. When this object is created a *context path* must be specified. This is a list of one or more package names that contain interfaces generated by the binding compiler.
  The following code snippet creates JAXBContext object whose context path is test.jaxb.

```
JAXBContext jc = JAXBContext.newInstance("test.jaxb");
```

- Create an *Unmarshaller* object. This object controls the process of unmarshalling. In particular, it contains methods that perform the actual unmarshalling operation.
  The following code snippet creates an Unmarshaller object:

```
import javax.xml.bind.Unmarshaller;

Unmarshaller unmarshaller = jc.createUnmarshaller();
```

- Call the Unmarshal method. This method does the actual unmarshalling of the XML document.
  The following statement unmarshals the XML data in the test.xml file.

```
Collection collection= (Collection) unmarshaller.unmarshal(new File( "test.xml"));
```

- Use the *get* methods in the schema-derived classes to access the XML data. The classes that a JAXB compiler generates for a schema include *get* and *set* methods that can be used to respectively obtain and specify data for each type of element and attribute in the schema.

### 3.1.3 JAXB for Public Contracts XML Files

Given the ease with which JAXB allows to access and process XML files, we decided to use this architecture to extract data from files containing information on public contracts that have been previously downloaded. To be able to process the data contained in the file, we carry out the binding of the schema and then the Unmarshal of the document as required. Before the binding we have slightly modified the original Schema provided by the Authority for the supervision of the public contracts.

More precisely, to be able to collect all the information from the analysed files, we have modified the elements of date type and decimal type as follow.

```
<xsd:sequence>
        <xsd:element name="dataInizio" type="xsd:date" minOccurs="0"/>
        <xsd:element name="dataUltimazione" type="xsd:date" minOccurs="0"/>
</xsd:sequence>
```

```
<xsd:sequence>
        <xsd:element name="dataInizio" type="legge190:dateType" minOccurs="0"/>
        <xsd:element name="dataUltimazione" type="legge190:dateType" minOccurs="0"/>
</xsd:sequence>

<xsd:simpleType name="dateType">
    <xsd:restriction base="xsd:string">
        <xsd:pattern value = "([0-9]{4})\-((0[13578])|(1[02]))\-((0[1-9])|(1[0-9])|(2[0-9])|3[01])"/>
        <xsd:pattern value = "([0-9]{4})\-(02)\-((0[1-9])|(1[0-9])|(2[0-9]))"/>
        <xsd:pattern value = "[0-9]{4}\-((0[469])|11)\-((0[1-9])|(1[0-9])|(2[0-9])|(30))"/>
    </xsd:restriction>
</xsd:simpleType>
```

All elements that are defined as date in XML schema are defined as string in the new XML used for the binding and the restrictions define the only valid value format, that is, YYYY-MM-DD as defined in original XML. This change is necessary because if a user enters a value that is out of domain, such as 10 march 2014, in this way we are able to collect it while,defining elements as date type, we would not be able to collect this out of domain information and it would not be manipulated in subsequent stage of transformation and it would be lost some of the fields for the evaluation of the quality. With the binding of the redefined XML Schema we are able to evaluate the quality of each element of the documents, having a 100% coverage of the Data Quality.

For the same reason, we changed the ImportoType as follow:

```
<xsd:simpleType name="ImportoType">
    <xsd:restriction base="xsd:decimal">
        <xsd:minInclusive value="0"/>
        <xsd:maxInclusive value="999999999999.99"/>
            <xsd:totalDigits value="15"/>
            <xsd:fractionDigits value="2"/>
    </xsd:restriction>
</xsd:simpleType>
```

```
<xsd:simpleType name="ImportoType">
        <xsd:restriction base="xsd:string">
            <xsd:pattern value = "[0-9]{0,15}"/>
            <xsd:pattern value = "\.[0-9]{1,2}"/>
            <xsd:pattern value = "[0-9]{1,12}\."/>
           <xsd:pattern value = "[0-9]{1,12}\.[0-9]{1,2}"/>
        </xsd:restriction>
</xsd:simpleType>
```

Therefore, the Contracting Authorities compile the files following the instructions on the structure and on the types of elements specified in the original XML Schema. We analyse the data generated by the Contracting Authorities, using the modified scheme as shown above.

From this Schema, we generate Java classes that represent it as shown in Figure 9.

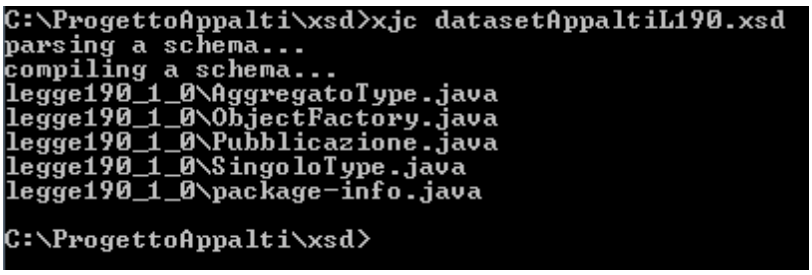



**Figure 9**: Results of the xjc command

There are generated classes corresponding elements defined in the schema. The class *package-info.java* is the package descriptor. The *ObjectFactory.java* class has the aim of allowing to create objects at run-time in response to user actions or of a flow of execution.

Once we have the classes we proceed with the unmarshalling of the document. We create an object tree which represents the contents and the structure of the document. The created objects are instances of the classes generated by the binding compiler.

Following the steps listed above we:

- Create the JAXBContext object, passing the context path, that is, the name of the package containing the generated classes.

```
String context = "generated.legge190_1_0";
JAXBContext jaxbContext;
jaxbContext = JAXBContext.newInstance(context);
```

- Create an Unmarshaller object.

```
Unmarshaller  unmarshaller = null;
unmarshaller = jaxbContext.createUnmarshaller();
```

- Perform the unmarshalling of all the xml documents stored in the directory specified by the path.

```
Object  objectJAXB = null;
objectJAXB=unmarshaller.unmarshal(new FileInputStream(filePath));
```

What is returned by the *unmarshal* method is a generic object, this object is then converted in the container class, that is the *Pubblicazione* class, by means of a cast operation. At the end of the unmarshalling we can access the content of the XML file by using the *get* methods and we can perform the second step of the extract, transform and load process.

## 3.2 Transformation Stage

In this phase, the data are processed to be adequately stored in specific structures for querying and analysis purposes. The extracted data are then converted into the form they need to be in so that they can be placed into a database. Before defining the transformation to apply on the data in order to make the analysis, we had to design a database capable of storing all the information extracted from the file and possibly modified for the evaluation of the different quality dimensions.

### 3.2.1 Data Model for Public Contracts

The creation of a database is a complex process because it is necessary to understand in depth the reality to model. A data model organizes data elements and standardizes how the data elements relate to one another. Since data elements documents real life people, places and things and the events between them, it represents reality. A data model establishes a convention to express different aspects of reality and constitutes a

support to its representation. The modelling phase is the most important in creating valid and well-performing databases, since an error made at this level, such as the forgetting of elements relation, can impact all subsequent stages.

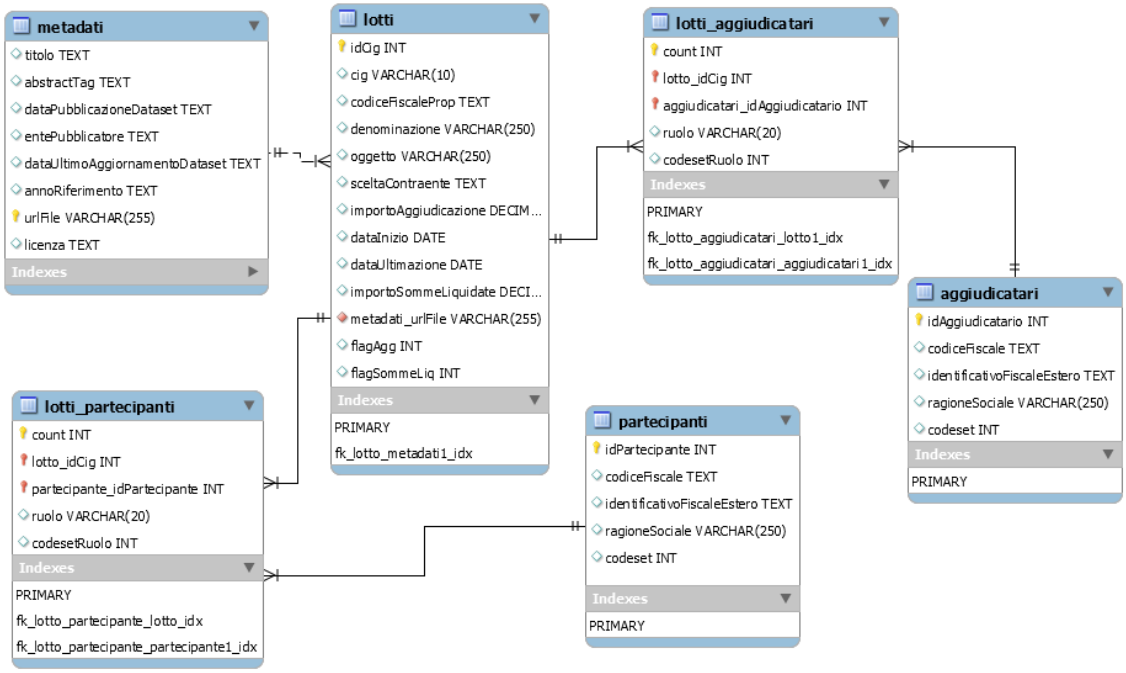



**Figure 10:** Public Contracts data model

In figure 10 is shown the model that we have created for storing and that we use for representing all the data extracted from the summary tables of public contracts. In that model are represented all the tables and the attributes of the database that we will use to perform the analysis. From the figure can be seen that, to simplify the structure of the database, table names and attribute names correspond to the tag names specified in the XML file. In the *metadati* tables there are the attributes corresponding to the elements that must be present in the metadata section of the document and the url of the file is used as the primary key. The *lotto*, *partecipanti* and *aggiudicatari* tables store all the information that are in the data section of the document. More precisely, all the data related to the lots are stored in the *lotti* table and we added two flag *flagAgg* and *flagSommeLiq* that will be valued during the transformation phase depending on the values of *importoAggiudicazione* and *importoSommeLiquidate* attributes and they will be useful in the analysis phase. The primary key of the *lotti* table is the *idCig* that is an auto incremental primary key associated to each lotto. We decided not to use the *cig* attribute as a primary key for two reasons. The first is that despite the cig should be unique, in some particular cases more than one cig can be zero. The second reason is that, as we said, the database is used to perform the

analysis of the data retrieved by the documents thus, during the transformation stage, to keep track of all the data errors that appear in the document, we assign particular values to the elements that are out of domain. Since it may happen that in the document are specified two or more out of domain cig, the value assigned after the transformation stage is the same for all of them and the cig would be not unique. The *urlFile* attribute of the *metadati* table is a foreign key of the *lotti* table, this is used to maintain the link among them. The *partecipanti* table is used to store the information of all the *participant* listed in a file in the *partecipanti* section while in the *aggiudicatari* table are stored all the information of each successful tenderer specified in the document in the *aggiudicatari* section and both of them use the same attributes. Even for the *partecipanti* and *aggiudicatari* tables the primary keys are incremental keys and we added in both the *codeset* flag that is valued during the transformation stage and is used to assess the quality of the *codiceFiscale* and *identificativoFiscaleEstero* attributes as explained in the next paragraph. To keep track of the participants taking part in each contract and the successful tenderer who win the contract itself, we use *lotti_partecipanti* and *lotti_aggiudicatari* tables. The key of the *lotti_partecipanti* table is defined by the triple *count*, *lotto_idCig,* and *partecipante_idPartecipante,* that is, a counter, the key of the lot in the *lotti* table (foreign key in *lotti_partecipanti*) and the key of the participant in the *partecipanti* table (foreign key in *lotti_partecipanti*). Moreover, if a participant takes part in more contracts this will be traced in that table. The key of the *lotti_aggiudicatari* table is given by the triple, *count*, *lotto_idCig*, *aggiudicatario_idAggiudicatario*, that is, a counter and the foreign keys *idCig* of *lotti* table and *idAggiudicatario* of *aggiudicatari*. We use the *lotti_aggiudicatari* table to track the successful tenderers for each contract and even if a successful tenderer is the winner of more than one contract. In both table there is the *ruolo* attributes that indicates, in the case the participant (or the successful tenderer depending on the table) is part of a group, the role of the participant in that group. The codeset flag is added for being able to perform the quality analysis on the *ruolo* attribute and is valued during the transformation step depending on the value of *ruolo* attribute.

With the introduced model we are able to represent the data in the XML file without losing any information contained in the summary tables regarding public contracts and this allows us to have 100% coverage of the Data Quality, that is, we will able to

evaluate the quality of each element provided by the public administration in the document.

### 3.2.2 Data Transformation

This phase consists on defining and implementing the transformation on the collected data in order to make them suitable for the quality assessment. The XML Schema, provided by the authorities, indicates the type of data to be entered. More precisely, the Schema gives indications on the data domain and the number of occurrences of each element. Using those information we define three cases in which transformation must be done. The cases evaluated for each tag are:

- Data retrieved is out of domain
- The tag element exists but is empty(<cig></cig>)
- The tag does not exist

Depending on the domain of the elements, we assign different values for the different cases. Before listing the transformations defined for the elements, it is necessary to know their domain as specified in the original XML Schema. We will show for each table in the database, the domain of the attributes as specified by the Schema and how we transform the data for the cases explained above.

**Lotto**

| Element | Source Domain |
|---|---|
| cig | String of exactly 10 characters. |
| codiceFiscaleProp | Numeric string of length 11. |
| denominazione | String of maximum length 250. |
| oggetto | String of maximum length 250. |
| sceltaContraente | Values specified by the sceltaContraenteType in the XML Schema. |
| importoAggiudicazione | Sequence of digits in European format with the constraint of 15 digits with 2 decimals. |
| dataInizio | Date with YYYY-MM-DD format. |
| dataUltimazione | Date with YYYY-MM-DD format. |
| importoSommeLiquidate | Sequence of digits in European format with the constraint of 15 digits with 2 decimals. |

| Attribute | Value Retrieved by the Document | Value assigned |
|---|---|---|
| cig | Out of domain | cig = NID |
| | Empty tag | cig = NID |
| | Tag not present | cig = null |
| codiceFiscaleProp | Out of domain | codiceFiscaleProp = NID |
| | Empty tag | codiceFiscaleProp = NID |
| | Tag not present | codiceFiscaleProp = null |
| denominazione | Out of domain | denominazione = NID |
| | Empty tag | denominazione = null |
| | Tag not present | denominazione = null |
| oggetto | Out of domain | oggetto = NID |
| | Empty tag | oggetto = null |
| | Tag not present | oggetto = null |
| sceltaContraente | Out of domain | sceltaContraente = NID |
| | Empty tag | sceltaContraente = NID |
| | Tag not present | sceltaContraente = null |
| importoAggiudicazione | Out of domain | importoAggiudicazione = 0,00 flagAgg = 1 |
| | Empty tag | importoAggiudicazione = 0,00 flagAgg = 1 |
| | Tag not present | importoAggiudicazione = 0,00 flagAgg = 2 |
| dataInizio | Out of domain | dataInizio=0001-01-01 |
| | Empty tag | dataInizio=0001-01-01 |
| | Tag not present | Not considered |
| dataUltimazione | Out of domain | dataUltimazione = 3999-12-31 |
| | Empty tag | dataUltimazione = 3999-12-31 |
| | Tag not present | Not Considered |
| importoSommeLiquidate | Out of domain | importoSommeLiquidate = 0,00 flagSommeLiq = 1 |
| | Empty tag | importoSommeLiquidate = 0,00 flagSommeLiq = 1 |
| | Tag not present | importoSommeLiquidate = 0,00 flagSommeLiq = 2 |

For the elements that have fixed length, such as *cig* and *codiceFiscaleProp*, we assign the value *NID* when the tag is present but is empty because in this case the length is equal to 0 and, thus, is different by the domain defined by the Schema while, for the elements for which is specified only the maximum length, we choose to assign a *null* value. The value assigned to *importoAggiudicazione* and *importoSommeLiquidate* is equal to *0.00* for the three considered cases, to distinguish them we associate *flagAgg* to *importoAggiudicazione* and *flagSommeLiq* to *importoSommeLiquidate* and these flags are then used for the analysis. For the *DataInizio* element we decided to assign the fixed value *0001-01-01* in the case in which the date inserted is out of domain or

if the tag exist but is blank while, for the same cases, we assign *3999-12-01* to *dataUltimazione*. We use these two dates because they won't collide with the integrity constraints defined on the dates in the analysis stage, that is they won't change the results of the evaluation as we will explain later. We do not consider the case in which the tag is not presence because the minimum number of occurrences fixed by the Schema is zero for both the dates.

**Partecipanti**

| Element | Domain |
|---|---|
| codiceFiscale | Numeric string of length 11 or 16. |
| codiceIdentificativoEstero | String. |
| ragioneSociale | String of maximum length 250. |

| Attribute | Value Retrieved by the Document | Value assigned |
|---|---|---|
| codiceFiscale | Out of domain | codiceFiscale= NID |
| | Empty tag | codiceFiscale = NID<br>codeset = See next table |
| | Tag not present | codiceFiscale = null<br>codeset = See next table |
| identificativoFiscaleEstero | Out of domain | Not Considered |
| | Empty tag | identificativoFiscaleEstero = null |
| | Tag not present | identificativoFiscaleEstero = null |
| ragioneSociale | Out of domain | ragioneSociale = NID |
| | Empty tag | ragioneSociale = null |
| | Tag not present | ragioneSociale = null |

| | codiceFiscale != null | codiceFiscale = null |
|---|---|---|
| identificativoFiscaleEstero=null | codeset = 1 | codeset = 4 |
| identificativoFiscaleEstero!=null | codeset =3 | codeset = 2 |

The *identificativoFiscaleEstero* as defined by the Schema must be a String, thus, the out of domain case in not considered because each string inserted is a correct one. The *codiceFiscale* and the *identificativoFiscaleEstero* cannot appear together for one participant, thus, when one of them is not present it is necessary to check if the other one is present or not. If both of them are not present is assigned to both the *null* value and the *codeset* is set to 4. The same check should be done if the tag of one of them is present but is empty. If the *codiceFiscale* tag is present and is empty but the *identificativoFiscaleEstero* is not present, an *NID* value is assigned to *codiceFiscale* and *codeset* is set to 1. On the contrary, if the *identificativoFiscaleEstero* tag is

present but is empty and the *codiceFiscale* is not present, a *null* value is assigned to it and the *codeset* is set to 2. If both, *codiceFiscale* and *identificativoFiscaleEstero*, are present the codeset is set to 3. The *codeset* will be then used for the quality evaluation. Since the *aggiudicatari* table has the same attributes of the *partecipanti*, the same transformations applied for the attribute of the *partecipanti* table are applied to the attributed of the *aggiudicatari* table.

**Lotti_partecipanti**

| Element | Domain |
|---|---|
| ruolo | Values specified by the ruoloType in the XML Schema. |

| Attribute | Value Retrieved by the Document | Value assigned |
|---|---|---|
| ruolo | Out of domain | ruolo = NID |
| | Blank tag | ruolo = null |
| | Tag not present | ruolo = null |

The *ruolo* tag must appear if and only if the participant is part of a group. If the participant is a member of a group, the *codeset* is set to 1 otherwise is 0. If the tag is not present the value assigned to *ruolo* is null but the *codeset* let us know if that participant is or not in a group. The *codeset* is used in next step to assess the quality. Since the *lotti_aggiudicatari* table has the same attributes of *the lotti_partecipanti,* the same transformations applied for the attribute of the *lotti_partecipanti* table are applied to the attributed of the *lotti_aggiudicatari* table.

## 3.3 Loading Stage

The loading stage is the last one and consists in storing the information, previously extracted and possibly transformed, in the database.

### 3.3.1 JDBC Database Access

The JDBC is a Java API that can access any kind of tabular data, especially data stored in Relational Database.

JDBC helps to write application that manages these three programming activities:

- Connect to a data source, like a database.
- Send queries and update statements to the database.
- Retrieve and process the results received from the database in answer to a query.

The following code fragment gives a simple example of these three steps:

```
Connection con = DriverManager.getConnection(
                    "jdbc:myDriver:myDatabase",
                     username,
                      password);

Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("SELECT a, b, c FROM Table1");

while (rs.next()) {
      int x = rs.getInt("a");
      String s = rs.getString("b");
      float f = rs.getFloat("c");
   }
```

This short code fragment instantiates a *DriverManager* object to connect to a database driver and log into the database, instantiates a *Statement* object that carries the SQL language query to the database; instantiates a *ResultSet* object that retrieves the results of the query, and executes a simple while loop, which retrieves and displays those results.

JDBC includes four components:

- The *JDBC API*: provides programmatic access to the relational data from the Java programming language. Using the JDBC API, applications can execute SQL statements, retrieve results and propagate changes back to an underlying data source. The JDBC can also interact with multiple data sources in a distributed, heterogeneous environment.
- *JDBC Driver Manager*: The JDBC *DriverManager* class defines objects which can connect Java applications to a JDBC Driver. *DriverManager* has traditionally been the backbone of the JDBC Architecture.
- *JDBC Test Suite*: The JDBC driver test suite helps to determine that JDBC drivers will run your program. These tests are not comprehensive or

exhaustive, but they do exercise many of the important features in the JDBC API.

- *JDBC-ODBC Bridge:* The Java Software bridge provides JDBC access via ODBC drivers.

JDBC API supports both two-tier and three-tier architectures processing models for database access. In the two tier architecture, java applet or application talks directly to the data source. This requires a JDBC driver that can communicate with the particular data source being accessed.

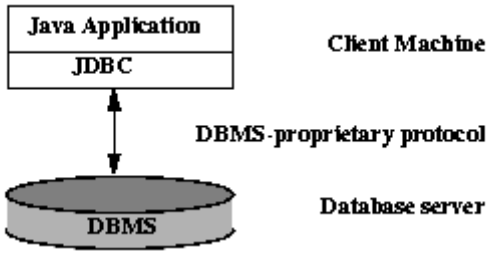



**Figure 11:** Two-tier Architecture for Data Access

User's commands are delivered to the database or other data source and the results of those statements are sent back to the user. The data source may be located on another machine to which the user is connected via a network. This is referred to as a client/server configuration, with the user's machine as the client, and the machine housing the data source as the server.

In the three tier architecture, commands are send to a "middle tier" of services, which then send the commands to the data source. The data source processes the commands and sends the results back to the middle tier, which then send them back to the user.

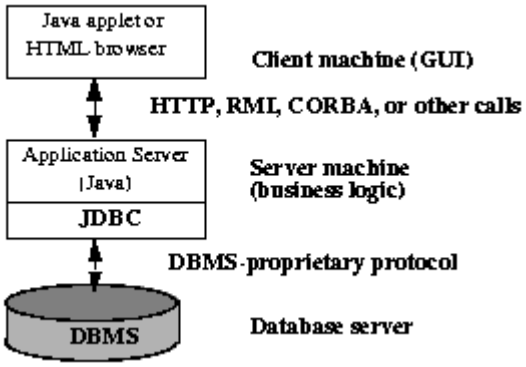



**Figure 12**: Three-tier Architecture for Data Access

### 3.3.1 JDBC Public Contracts Database Access

Given the easy with which it accesses databases, we use the JDBC API to load the information about the public contract.

To insert the extracted and transformed data, the main steps are:

- Instantiate the *DriverManager* to connect to the database

```
String url = "jdbc:mysql://localhost:3306/";
String dbName = "appalti";
String driver  = "com.mysql.jdbc.Driver";
String userName = username;
String password = password;

Class.forName(driver).newInstance();
Connection conn =
DriverManager.getConnection(url+dbName,userName,password);
```

- Create and SQL INSERT statement, using the *PreparedStatement* syntax.

```
PreparedStatement pstm;

pstm = conn.prepareStatement("INSERT INTO
appalti.metadati(titolo,abstractTag,dataPubblicazioneDataset,enteP
ubblicatore,dataUltimoAggiornamentoDataset,annoRiferimento,urlFi
le,licenza) VALUES
('"+titolo+"','"+abstractTag+"','"+dataPubblicazioneDataset+"','"+ente
Pubblicatore+"','"+dataUltimoAggiornamentoDataset+"','"+annoRifer
imento+"','"+urlFile+"','"+licenza+"')");
```

- Executes a Java *PreparedStatement*.

```
pstm.execute();
```

- Close the Java MYSQL database connection.

```
conn.close();
```

In the code snippets, we have shown how insert the data in the metadata table but this procedure it is used to load all the data retrieved by the document in the appropriate tables.

To perform the insert statement it is convenient use the *PreparedStatement* object for sending the SQL statement to the database. This special type of Statement is derived from the more general class, *Statement.* If a *Statement* object has to be executed many times, it usually reduces execution time to use a *PreparedStatement* object instead. The main feature of a *PreparedStatement* object is that, unlike a *Statement* object, it is given a SQL statement when it is created. The advantage to this is that in most cases, this SQL statement is sent to the DBMS right away, where it is compiled. As a result, the *PreparedStatement* object contains not just a SQL statement, but a SQL statement that has been precompiled. This means that when the *PreparedStatement* is executed, the DBMS can just run the *PreparedStatement* SQL statement without having to compile it first. Since we perform a big amount of insert operation, we choose to use the *PreparedStatement*.

At the end of the ETL process, the data were extracted from the source file, if necessary they are transformed as explained in the previous section and loaded into the database. At this point we have a database that contains all the information and the relationships among the data extracted from files and we can proceed with the evaluation of such data.

*Chapter 4*

# 4. Data Quality

Organizations run on data. They use it to manage programs, select products to develop, make decisions, and guide improvement.

Data are generally considered high quality if "they are fit for their intended uses in operations, decision making, and planning"(J.M. Juran). This definition implies that data quality is both a subjective perception of individuals involved with the data, as well as the quality associated with the objective measurements based on the data set in question. The data quality is an essential characteristic that determines the reliability of data for decision making process.

The data quality is important for many reasons. The first is that the lack of quality increases the costs, the report on data quality of the Data Warehouse Institute estimates that the quality problems cost U.S. businesses more than 600 billion dollars a year (Data quality – Concept, Methodologies and Techniques. Batini & Scannapieca). The second reason is the data quality can be improved and this leads to a greater competitive advantage.

The quality problem is on one hand a permanent problem and on another is a problem in continuous evolution. It is permanent because each complex organization faces with the quality of information generated and used in its decision-making processes while, it is a problem that evolves constantly because its importance increases with the complexity of the organization, the size of the flows of communication and the distance between production and use of data. Furthermore, quality requirements vary over time and are thus subject to evolution.

If we imagine the government and citizens, respectively, as an organization, and customers we can get the importance of data quality not only for the private organizations but also for public administrations. The government's role is to guide the development through a decision-making process with the aim to increase citizens satisfaction. The government provides data and uses that data to make decision. The data made available by public administrations play a fundamental role for two reasons. The first is that they can be reused promoting the economic development, the

second is that they increase the transparency with consequent reduction of the illegality. The data are therefore fundamental in all relationships between governments, businesses and citizens. Public administrations are faced with different problems on the data due to their poor quality. For example, some recurring problems are that similar information about one citizen or business is likely to appear in multiple databases because each database is autonomously managed by the different agencies that historically has never been able to share data about citizens and businesses. The problem is worsened by the many errors usually present in the databases, for many reasons. First, due to the nature of the administrative flows, several citizens' data (e.g. addresses) are not updated for long periods of time. Also, errors may occur when personal data on individuals is stored. Some of these errors are not corrected and a potentially large fraction of them is not detected. Furthermore, data provided by distinct sources differ in format, following local conventions, that can change in time and result in multiple versions. Finally, many of the records currently in the database were entered over years using legacy processes that included one or more manual data entry steps. A direct consequence of this combination of redundancy and errors in data, is frequent mismatches between different records that refer to the same citizen or business. One major outcome of having multiple disconnected views for the same information is that citizens and businesses experience consistent service degradation during their interaction with the administrations. Furthermore, misalignment brings about additional costs. First, administrations must make an investment to reconcile records secondly, because most investigation techniques, e.g., tax fraud prevention techniques, rely on cross-referencing records of different agencies, misalignment results in undetected tax fraud and reduced revenues and transparency. Thus, to fully exploit the potential of the data, both those produced by public administrations and private organizations, it is essential that they have a high level of quality.

## 4.1 State of the art of Data Quality Models

### 4.1.1 Data Quality Models

When people thinks about data quality, they often reduce it to the accuracy but it is more than the simple accuracy. Other significant dimensions such as completeness, consistency and currency are necessary to fully characterize the quality of data. What is needed to evaluate the quality is an approach to methodically put in place data

quality measures applicable on the data. The most of the frameworks for the quality of the data are part of methodologies for their improvement.

A model of data quality, specifies:

- Definition of data quality.
- Dimensions of data quality.
- Metrics.
- How to control and improve data quality.

Data quality is considered important not only within the open data but in the evaluation of the data in general, for this purpose the models used for the evaluation of the general data are adapted to the open data.

This section provides an overview of the most used models and metrics that compose them. In Figure 13 are described the most used models while in Figure 14 the most used metrics. In the Figure 15 is specified from which metrics the models are composed.

| Methodology Acronym | Extended Name | Main Reference |
|---|---|---|
| TDQM | Total Data Quality Management | Wang 1998 |
| DWQ | The Datawarehouse Quality Methodology | Jeusfeld et al. 1998 |
| TIQM | Total Information Quality Management | English 1999 |
| AIMQ | A methodology for information quality assessment | Lee et al. 2002 |
| CIHI | Canadian Institute for Health Information methodology | Long and Seko 2005 |
| DQA | Data Quality Assessment | Pipino et al. 2002 |
| IQM | Information Quality Measurement | Eppler and Münzenmaier 2002 |
| ISTAT | ISTAT methodology | Falorsi et al 2003 |
| AMEQ | Activity-based Measuring and Evaluating of product information Quality (AMEQ) methodology | Su and Jin 2004 |
| COLDQ | Loshin Methodology (Cost-effect Of Low Data Quality | Loshin 2004 |
| DaQuinCIS | Data Quality in Cooperative Information Systems | Scannapieco et al. 2004 |
| QAFD | Methodology for the Quality Assessment of Financial Data | De Amicis and Batini 2004 |
| CDQ | Comprehensive methodology for Data Quality management | Batini and Scannapieco 2006 |

**Figure 13:** Most used models (provided by Batini & Cappiello , Methodologies for Data Quality Assessment and Improvement, 2009, p. 12)

| Dimensions | Name | Metrics Definition |
|---|---|---|
| Accuracy | Acc1 | Syntactic accuracy: it is measured as the distance between the value stored in the database and the correct one<br>Syntactic Accuracy=Number of correct values/number of total values |
| | Acc2 | Number of delivered accurate tuples |
| | Acc3 | User Survey - Questionnaire |
| Completeness | Compl1 | Completeness = Number of not null values/total number of values |
| | Compl2 | Completeness = Number of tuples delivered/Expected number |
| | Compl3 | Completeness of Web data = $(T_{max}- T_{current})^*$ $(Completeness_{Max}- Completeness_{Current})/2$ |
| | Compl4 | User Survey - Questionnaire |
| Consistency | Cons1 | Consistency = Number of consistent values/number of total values |
| | Cons2 | Number of tuples violating constraints, number of coding differences |
| | Cons3 | Number of pages with style guide deviation |
| | Cons4 | User Survey - Questionnaire |
| Timeliness | Time1 | Timeliness = $(\max(0; 1\text{-Currency/Volatility}))^s$ |
| | Time2 | Percentage of process executions able to be performed within the required time frame |
| | Time3 | User Survey - Questionnaire |
| Currency | Curr1 | Currency = Time in which data are stored in the system - time in which data are updated in the real world |
| | Curr2 | Time of last update |
| | Curr3 | Currency = Request time- last update |
| | Curr4 | Currency = Age + (Delivery time- Input time) |
| | Curr5 | User Survey - Questionnaire |
| Volatility | Vol1 | Time length for which data remain valid |
| Uniqueness | Uni1 | Number of duplicates |
| Appropriate amount of data | Appr1 | Appropriate Amount of data = Min ((Number of data units provided/Number of data units needed); (Number of data units needed/Number of data units provided)) |
| | Appr2 | User Survey - Questionnaire |
| Accessibility | Access1 | Accessibility = max (0; 1-(Delivery time - Request time)/(Deadline time - Request time)) |
| | Access2 | Number of broken links - Number of broken anchors |
| | Access3 | User Survey - Questionnaire |
| Credibility | Cred1 | Number of tuples with default values |
| | Cred2 | User Survey - Questionnaire |
| Interpretability | Inter1 | Number of tuples with interpretable data, documentation for key values |
| | Inter2 | User Survey - Questionnaire |
| Usability | Usa1 | User Survey - Questionnaire |
| Derivation Integrity | Integr1 | Percentage of correct calculations of derived data according to the derivation formula or calculation definition |
| Conciseness | Conc1 | Number of deep (highly hierarchic) pages |
| | Conc2 | User Survey - Questionnaire |
| Maintainability | Main1 | Number of pages with missing meta-information |
| Applicability | App1 | Number of orphaned pages |
| | App2 | User Survey - Questionnaire |
| Convenience | Conv1 | Difficult navigation paths: number of lost/interrupted navigation trails |
| Speed | Speed1 | Server and network response time |
| Comprehensiveness | Comp1 | User Survey - Questionnaire |
| Clarity | Clar1 | User Survey - Questionnaire |
| Traceability | Trac1 | Number of pages without author or source |
| Security | Sec1 | Number of weak log-ins |
| | Sec2 | User Survey - Questionnaire |
| Correctness | Corr1 | User Survey - Questionnaire |
| Objectivity | Obj1 | User Survey - Questionnaire |
| Relevancy | Rel1 | User Survey - Questionnaire |
| Reputation | Rep1 | User Survey - Questionnaire |
| Ease of operation | Ease1 | User Survey - Questionnaire |
| Interactivity | Interact1 | Number of forms - Number of personalizable pages |

**Figure 14:** Description of the metrics (provided by Batini & Cappiello , Methodologies for Data Quality Assessment and Improvement, 2009, p. 19)

| | TDQM | DWQ | TIQM | AIMQ | CIHI | DQA | IQM | ISTAT | AMEQ | COLDQ | DaQuinCIS | QAFD | CDQ | #metr/#dim |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Acc1 | X | | X | | | X | | X | X | X | X | X | X | 9/13 |
| Acc2 | | X | | | | | | | | | | | | 1/13 |
| Acc3 | | | | X | X | | X | | | | | | | 2/13 |
| Compl1 | X | | | X | | X | | | X | X | X | x | x | 7/12 |
| Compl2 | | X | X | | | | | | | | | | | 2/12 |
| Compl3 | | | | | | | | | | | | | X | 1/12 |
| Compl4 | | | X | | | | | | | | | | | 2/12 |
| Cons1 | X | | | | | X | | | | | | | | 6/10 |
| Cons2 | | X | | | | | | | | | | | | 1/10 |
| Cons3 | | | | | | | X | | | | | | | 1/10 |
| Cons4 | | | | X | X | | | | | | | | | 2/10 |
| Time1 | | | | | | X | | | | X | | | X | 3/7 |
| Time2 | | | X | | | | | | | X | | | | 2/7 |
| Time3 | | | | X | X | | | | | | | | | 2/7 |
| Curr1 | | | | | | | | X | | | X | | | 2/8 |
| Curr2 | X | | | | | | X | | | | | X | | 2/8 |
| Curr3 | | | | X | X | | | | | | | | | 1/8 |
| Curr4 | | | | | | | | | | | | | X | 1/8 |
| Curr5 | | | | X | X | | | | | | | | | 2/8 |
| Vol1 | X | | | | | | | | | | | | X | 2/2 |
| Uni1 | | | X | | | | | | | | | X | | 1/2 |
| Appr1 | | | | | | X | | | | | | | | 1/2 |
| Appr2 | | | | X | | | | | | | | | | 1/2 |
| Access1 | | | | | | | X | | | | | | | 1/4 |
| Access2 | | | | | | | | X | | | | | | 1/4 |
| Access3 | | | X | X | | | | | | | | | | 2/4 |
| Cred1 | | X | | | | | | | | | | | | 1/2 |
| Cred2 | | | | X | | | | | | | | | | 1/2 |
| Inter1 | | X | | | | | | | | | | | | 1/2 |
| Inter2 | | | | X | | | | | | | | | | 1/2 |
| Usa1 | | | X | | | | | | | | | | | 1/1 |
| Integr1 | | | X | | | | | | | | | | | 1/1 |
| Conc1 | | | | | | | | | | | | | | 1/2 |
| Conc2 | | | | | | | X | | | | | | | 1/ 2 |
| Main1 | | | | X | | | | | | | | | | 1/1 |
| App1 | | | | | | | | | | | | | | 1/1 |
| App2 | | | | | | | X | | | | | | | 1/1 |
| Conv1 | | | | | | | X | | | | | | | 1/1 |
| Speed1 | | | | | | | X | | | | | | | 1/1 |
| Comp1 | | | | X | | | X | | | X | | | | 3/3 |
| Clar1 | | | X | | | | X | | | X | | | | 3/3 |
| Trac1 | | | | | | | X | | | | | | | 1/1 |
| Sec1 | | | | | | | X | | | | | | | 1/1 |
| Sec2 | | | | X | | | | | | | | | | 1/1 |
| Corr1 | | | | | | | X | | | | | | | 1/1 |
| Obj1 | | | | X | | | | | | | | | | 1/1 |
| Rel1 | | | | X | | | | | | | | | | 1/1 |
| Rep1 | | | | X | | | | | | | | | | 1/1 |
| Ease1 | | | | X | | | | | | | | | | 1/1 |
| Interact1 | | | | | | | X | | | | | | | 1/1 |

**Figure 15:** Metrics defined by each model (provided by Batini & Cappiello , Methodologies for Data Quality Assessment and Improvement, 2009, p. 20)

### 4.1.2 Open Data Quality Models

Tim Berners – Lee, the inventor of the web and Linked Data initiator, suggested a *5 star deployment scheme* for Open Data. The levels are identified by the stars that are assigned to the data set according to the way and the format in which it is published.

- *1 star*: Is assigned when the data are available on web under an Open License.
- *2 stars*: are assigned to structured data such as Excel.

- *3 stars*: are assigned to data that are made available in a non-proprietary open format such as csv.
- *4 stars*: are assigned when the URI of the data is specified so that people can point to them.
- *5 stars*: are assigned to data that are linked to other data to provide context.

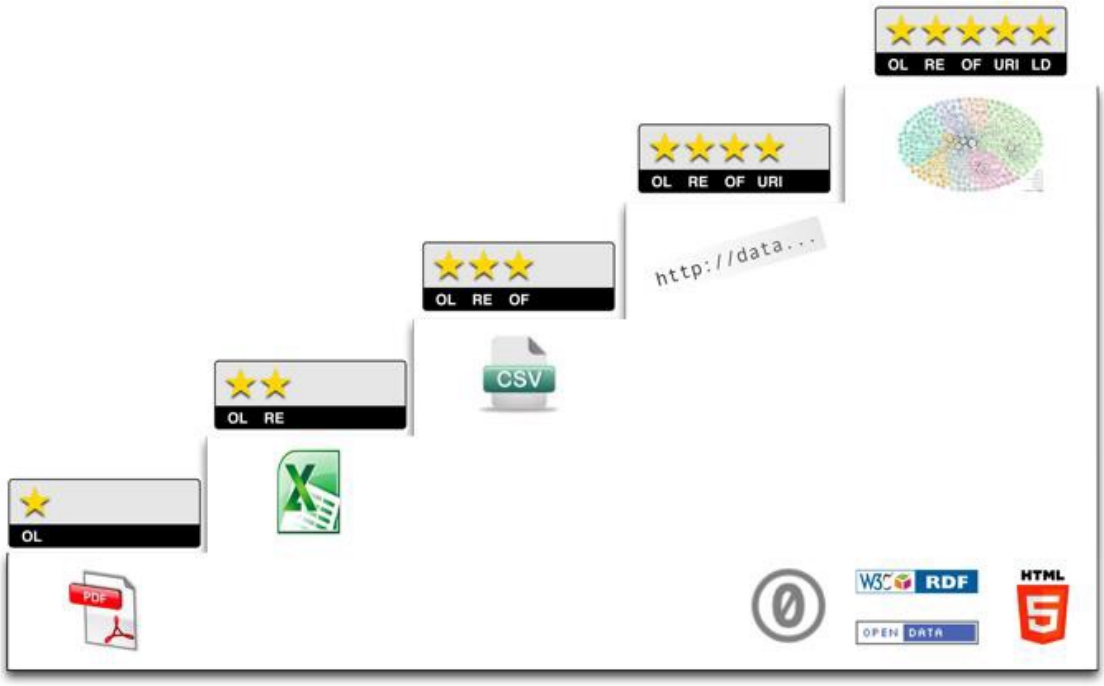



**Figure 16**: Five star open data(FSOD)

Although the model is simple and intuitive, it can only be used for a first assessment of the data set that has to be analysed and to assess the level of the portals that provide open data and it does not give any information about the quality of the data. To achieve that objective, the model SPODQM (Orozco, Torchiano, & Vetrò, 2013) can be used. Such a model, for the evaluation of the open data portals, is based on the standards SQuaRE (ISO/IEC 25012), the PDQM model (Calero, Caro, & Piattini, 2008) and SPDQM (Moraga,2009) which combines SQuaRE and PDQM. The quality model defined by the standards SQuaRE consists of 15 features that sum up those that have positive impact on quality. These features can be grouped in two different groups that are:

- *Inherent*: It indicates that the data itself can meet the needs of users when they are used in a specific context.

- *System dependent*: It includes features which indicate that the quality of the data is guaranteed within a specific computer system and when the data are used in a specific context.

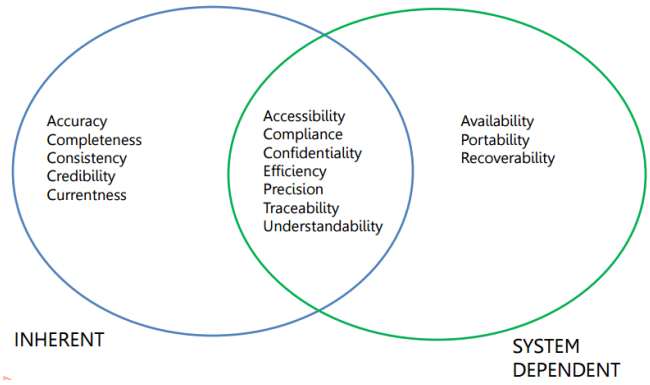



**Figure 17:** ISO/IEC 25012 – SQUARE

The PDQM (Data Quality models for web portals) is used as model for the evaluation of web portals from the user point of view. It provides a set of useful features for user evaluation of the data on web portals. These features can be grouped in four categories:

- *Intrinsic:* It indicates the aspects of quality depending on data itself.
- *Operational:* it includes dimensions which depend on the system.
- *Contextual:* it includes the aspect that may be considered depending on the context.
- *Representational:* It includes aspects related to the representation of the data.

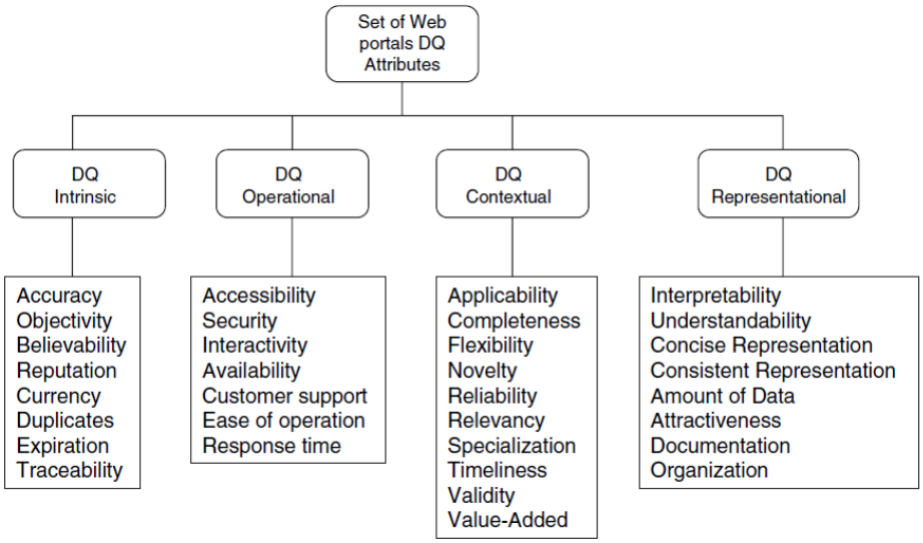



**Figure 18:** PDQM: Portal Data Quality Model

The SPDQM combines together the standard ISO/IEC 25012 and the PDQM to provide an easy an flexible model used for the quality evaluation of any web portal. It mixes the features of the two models and defines 42 features which are grouped into:

- *Categories*: it inherits the categories defined by the PDQM model.
- *Points of view:* it inherits the group defined in the ISO/IEC 25012.

To assess the quality of the open data, in the studies conducted by Vetrò and Torchiano (Vetrò, Torchiano, & Orozco, 2014), it is proposed the SPODQM. This model is divided in different level which are inherited from the SPDQM and are also evaluated some of the important aspects of Open Data. The first level is the point of view which include the categories that in turn include the characteristics which contain the sub – characteristics that are showed in figure 19.

| Point of view | Category | Characteristic | Sub Characteristic |
|---|---|---|---|
| Inherent | Intrinsic: this denotes that data have quality in their own right | Accuracy | |
| | | Credibility | Objectivity |
| | | | Reputation |
| | | Traceability | |
| | | Currentness | |
| | | Expiration | |
| | | Completeness | |
| | | Consistency | |
| | | Accessibility | |
| | | Compliance | |
| | | Confidentiality | |
| | | Efficiency | |
| | | Precision | |
| | | Understandability | |
| System dependent | Operational: this emphasizes the importance of the role systems; that is, the system must be accessible but secure | Availability | |
| | | Accessibility: | Interactive |
| | | | Ease of operation |
| | | | Customer Support |
| | | Verifiability | |
| | | Confidentiality | |
| | | Portability | |
| | | Recoverability | |
| | Contextual: this highlights the requirement which states that data quality must be considered within the context of the task in hand | Validity: | Reliability |
| | | | Scope |
| | | Value-added: | Applicability |
| | | | Flexibility |
| | | | Novelty |
| | | Relevancy: | Novelty |
| | | | Timeliness |
| | | Specialization | |
| | | Usefulness | |
| | | Traceability | |
| | | Compliance | |
| | | Precision | |
| | Representational: this denotes that the system must present data in such a way that they are interpretable, easy to understand, and concisely and consistently represented | Concise Representation | |
| | | Consistent Representation | |
| | | Understandability | Interpretability |
| | | | Amount of data |
| | | | Documentation |
| | | | Organization |
| | | Attractiveness | |
| | | Readability | |
| | | Efficiency | |
| | | Effectiveness | |

**Figure 19:** SPDQM Model

## 4.2 Choice of Framework

After briefly introduced models for the quality of the data, we propose the model we use for the evaluation of public contracts data. Studying the various dimensions, we decided not to use any of the models described above. The evaluation is performed using what can be considered an hybrid model in that it takes into account some intrinsic characteristics of the data, that is, dimensions that can be evaluated for any data type and some features that strictly depend on the type of data, that is, public contracts. In the following are described the different dimensions and the main metrics for their evaluation. It is also provided the definitions of dimension that best fit the data taken into account and which will form the model for the evaluation of public contracts.

### 4.2.1 Accuracy

*Accuracy* is defined as the closeness between a value *v* and a value *v'*, considered as the correct representation of the real-life phenomenon that *v* aims to represent. Two kinds of *accuracy* can be identified, namely a *syntactic accuracy* and a *semantic accuracy*.

*Syntactic accuracy:* is the closeness of a value *v* to the elements of the corresponding definition domain *D*. In *syntactic accuracy* we are not interested in comparing *v* with the true value *v'*; rather, we are interested in checking whether *v* is any one of the values in *D*, whatever it is. It is measured by means of functions, called *comparison function,* that evaluate the distance between *v* and the values in *D*.

A metric to calculate this kind of *accuracy* could be:

$$Q_a = 1 - d(v, D)$$

with:

$$d(v, D) = \frac{distance}{\max distance}$$

where:

*distance:* minimum distance so that the value is within the domain.

*max distance*: It indicates how many characters should be changed if they were all wrong.

This metric can be calculated on single value or with a higher aggregation on the columns and tables.

A simpler metric, but which provides less information, to compute the distance among *v* and the domain *D* is:

$$d(v, D) = 0 \text{ if } v \epsilon D, 1 \text{ otherwise}$$

This metric is understandable to non-experts users since evaluating at dataset level, the metric is:

$$Q_a(v, D) = 1 - (\frac{Total\ number\ of\ values\ belonging\ to\ the\ domain}{Total\ number\ of\ data})$$

*Semantic Accuracy:* is the closeness of the value *v* to the true value *v'*. While it is reasonable to measure *syntactic accuracy* using a distance function, *semantic accuracy* is measured better with a <yes, no> or a <correct, not correct> domain. Consequently, *semantic accuracy* coincides with the concept of *correctness*. In order to measure the *semantic accuracy* of a value *v*, the corresponding true value has to be known, or, else, it should be possible, considering additional knowledge, to deduce whether the value *v* is or is not the true value.

The accuracy when measured with a binary variable can be calculated with the formula:

$$correctness = 1 - (\frac{Total\ number\ of\ errors}{Total\ number\ of\ data})$$

Generally, for relation and database accuracy, for both *syntactic* and *semantic accuracy*, a *ratio* is typically calculated between accurate values and the total number of values. For instance, the *accuracy* of a relation can be measured as the ratio between the number of correct cell values and the total number of cells in the table.

For the specific case of public contracts, given the impossibility of having a dataset that represents all possible values without errors to compare with the original data, it is impossible to establish the *correctness* of the data. We exploit the information about the domain of the data provided by the XML Schema, to evaluate if values belong to the domain, that is, we assess the *syntactic accuracy* of the data.

### 4.2.2 Completeness

*Completeness* can be generically defined as:

*"the extent to which data are of sufficient breadth, depth, and scope for the task at hand*"(Wang & Strong, 1996).

According to the level which the data are analysed, three types of *completeness* can be defined:

- *Schema completeness:* is defined as the degree to which concepts and their properties are not missing from the schema.
- *Column completeness*: is defined as a measure of the missing values for a specific property or column in a table.
- *Population completeness*: evaluates missing values with respect to a reference population.

Moreover, the *completeness* in the relational schema characterizes the extent to which the table represents the corresponding real world. *Completeness* in the relational model can be characterized with respect to (Data Quality – Concept, Methodologies and Techniques. Batini & Scannapieca):

- Presence or absence and meaning of the null values.
- The validity of one of the two assumptions called *open world assumption* and *closed world assumption.*

In a model with null values, the presence of a null value has the general meaning of missing value. In order to characterize completeness it is important to understand why the value is missing. Indeed, a value can be missing either because it exists but is unknown, or because it does not exist at all, or because it may exist but it is not actually known whether it exists or not.

In logical models for databases, such as the relational model, there are two different assumptions on the completeness of data represented in a relation instance r. The *closed world assumption* (CWA) states that only the values actually present in a relational table r, and no other values represent facts of the real world. In the *open world assumption* (OWA) we can state neither the truth nor the falsity of facts not represented in the tuples of r.

From the four possible combinations emerging from considering or not considering null values, and OWA and CWA, we will focus on the following two most interesting cases:

- Model without null values with OWA.
- Model without null values with CWA.

To characterize the *completeness* in a model without null values with OWA we should have a *reference relation ref(r),* for the relation *r,* containing all the tuples that satisfy the relational schema of *r.* In practical situations, the reference relations are rarely available. Instead their cardinality is much easier to get, thus, it is easier to comput the *completeness* of the relation r as the fraction of tuples actually represented in the relation *r*, namely, its size with respect to the total number of tuples in *ref(r):*

$$Completeness = \frac{|r|}{|ref(r)|}$$

In the model with null values with CWA, specific definitions for completeness can be provided by considering the granularity of the model elements:

- *Value completeness:* to capture the presence of null values for some fields of a tuple.
- *Tuple completeness:* to characterize the completeness of a tuple with respect to the values of all its fields.
- *Attribute completeness:* to measure the number of null values of a specific attribute in a relation.
- *Relation completeness:* to capture the presence of null values in a whole relation.

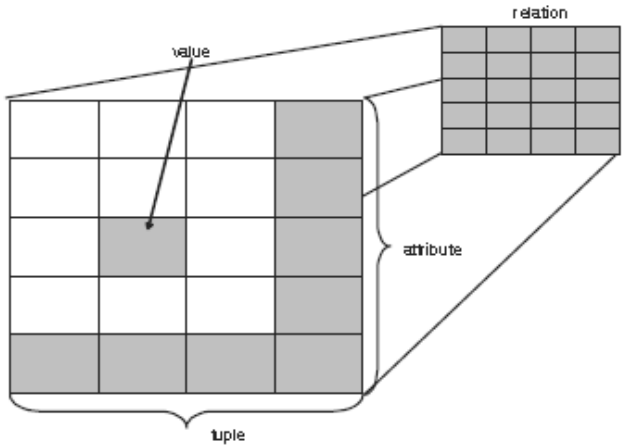



**Figure 20:** Completeness of different elements in the relational model(Provided by Data Quality – Concept, Methodologies and Techniques. Batini & Scannapieca)

The metric that could be used for the model with null values with CWA is:

$$Q_c(record) = 1 - \frac{\sum_{i=1}^{n}[field_i = null]}{n}$$

This metric is very intuitive but has the disadvantage of treating all fields with the same importance. The weight of certain attributes depends on the context, not all of the attributes may be relevant in the same way. A solution to this problem would be to weigh each attribute, however, assigning weight to each single attribute it is a task that must be carried out by an expert. The weight is assigned based on the importance of the attributes, an idea is to assign a weight 1 to attributes with a medium importance, the value 3 to the important attributes and the value 0 to the attributes that aren't important. The metric of *completeness* becomes(Reiche & Hofig, 2013):

$$Q_w(tuple) = \frac{\sum_{i=1}^{n} w_i\,[field_i \neq null]}{\sum_{i=1}^{n} w_j}$$

For the case of public contracts the model without null values and with OWA would provide a high accuracy in the evaluation of the *completeness* but cannot be used since we do not have a reference relation. For this reason we use the model without

null values and CWA and in particular we evaluate the tuples and attributes *completeness*. We also decided not to assign different weights to different attributes as a wrong assignment would lead to an incorrect assessment of *completeness*.

### 4.2.3 Duplication

When considering *accuracy* for sets of values instead of single values, a further notion of *accuracy* can be introduced, namely *duplication*. *Duplication* occurs when a real-world entity is stored twice or more in a data source. Of course, if a primary key consistency check is performed when populating a relational table, a duplication problem does not occur if the primary key assignment has been made with a reliable procedure. The *duplication* problem is more relevant for files or other data structures that do not allow the definition of key constraints.

The *duplication* of data related to public contracts is evaluated on the values that if duplicated would make the provided data unreliable.

### 4.2.4 Time-Related Dimensions: Currency, Timeliness and Volatility

An important aspect of data is their change and update in time. The classification of types of data can be performed according to the temporal dimensions, in terms of stable, long-term changing and frequently changing data. The principal time-related dimensions proposed for characterizing the three types of data are *currency*, *volatility*, and *timeliness*.

*Currency* concerns how promptly data are updated. *Currency* can be typically measured with respect to *last update* metadata, which correspond to the last time the specific data were updated. For data types that change with a fixed frequency, last update metadata allow us to compute *currency* straightforwardly. Conversely, for data types whose change frequency can vary, one possibility is to calculate an average change frequency and perform the *currency* computation with respect to it, admitting errors:

$$currency = age + (delivery\ date - input\ date)$$

where:

*age*: it indicates how old the data is when received.

*delivery date*: is when the data is delivered(Actual date for web data).

*input date*: it indicates when the data is obtained.

Therefore, *currency* is the sum of how old data are when received (*Age*), plus a second term that measures how long data have been in the information system, (*DeliveryTime − InputTime*).

Although data on public contracts specify the information about the last update, we cannot define how often they are changed. For this reason, in the development of our model we does not consider the *currency* as a dimension to assess.

*Volatility* characterizes the frequency with which data vary in time. It is a dimension that inherently characterizes certain types of data. Stable data have *volatility* equal to zero. A metric for *volatility* is given by the length of time that data remain valid (Batini & Scannapieca,2006).

$$volatility = range\ of\ time\ in\ which\ data\ remain\ valid$$

Since it is not possible to indicate precisely the range of time in which the data relating to public contracts remain valid, we decided not to consider this dimension.

*Timeliness* expresses how current data are for the task at hand. The *timeliness* dimension is motivated by the fact that it is possible to have current data that are actually useless because they are late for a specific usage. *Timeliness* implies that data not only are current, but are also in time for events that correspond to their usage. Therefore, a possible measurement consists of:

- a *currency* measurement.
- a check that data are available before the planned usage time.

*Timeliness* is defined as (Batini & Scannapieca,2006):

$$timeliness = \max\{0,1 - \frac{currency}{volatility}\}$$

Since we are not able to evaluate the *currency* and *volatility* for public contracts, we do not include the *timeliness* evaluation in our model.

### 4.2.3 Consistency

The consistency dimension captures the violation of semantic rules defined over a set of data items, where items can be tuples of relational tables or records in a file. With reference to relational theory, an instantiation of such semantic rules are called *integrity constraints.*

#### 4.2.3.1 Integrity Constraints

*Integrity constraints* are properties that must be satisfied by all instances of a database schema. Although *integrity constraints* are typically defined on schemas, they can at the same time be checked on a specific instance of the schema that presently represents the extension of the database.

It is possible to distinguish to main categories of *integrity constraints*:

- *Intrarelational Integrity Constraints:* regard single attribute or multiple attributes of a relation.
- *Interrelational Integrity Constraint:* involve attributes of more than one relation.

A metric that could be used for the evaluation of the consistency for the *intrarelational integrity constraints* is:

$$consistency = \frac{number\ of\ attributes\ which\ do\ not\ respect\ the\ constraint}{total\ number\ of\ tuple}$$

The data for public contracts allow the definition of many constraints. Some of them are specified in the XML Schema some other are logical constraints that we have

defined on the data after a careful observation of them. In the evaluation of the data we have defined both *intrarelational constraints* and *interrelational constraints* as defined in the next paragraph.

## 4.2 Model for Public Contract Assessment

At the end of this analysis we have a wide knowledge of the dimensions and of some metrics used for the evaluation of the data and that are used to build the model to assess the public contracts. The final model is, as we said earlier, an hybrid model which combines both the intrinsic dimensions such as *accuracy* and *completeness* and dimensions which are specifically defined for public contracts as the *consistency* with the definition of *integrity constraints*.

The dimensions that create the model for the evaluation of public contracts are given in figure 20.

| Dimension | Acronym | Metric |
|---|---|---|
| Accuracy | pcvc | Percentage of cells with correct value(value belonging to the domain). |
| Completeness | pcc | Percentage of complete cells. |
| | pcrp | Percentage of complete tuples. |
| Duplication | dup_participant | Number of participant which are duplicated in each lot. |
| | dup_tenderer | Number of successful tenderer which are duplicated in each lot. |
| Consistency | cf_ife_partecipanti | Percentage of tuples that meet the following Intrarelational Constraint:<br>- *codiceFiscale* and *identificativoFiscaleEstero* in the *partecipanti* table must not be simultaneously not null. |
| | cf_ife_aggiudicatari | Percentage of tuples that meet the following Intrarelational Constraint:<br>- *codiceFiscale* and *identificativoFiscaleEstero* in the *aggiudicatari* table must not be simultaneously not null. |
| | cf_eq_zero_partecipanti | Percentage of cells that meet the following Intrarelational Constraint:<br>- *codiceFiscale* in *partecipanti* table must be different by zero. |

| | | |
|---|---|---|
| | cf_eq_zero_aggiudicatari | Percentage of cells that meet the following Intrarelational Constraint:<br>- *codiceFiscale* in *aggiudicatari* table must be different by zero. |
| | check_on_date | Percentage of tuples that meet the following Intrarelational Constraint:<br>- *dataInizio* must be less recent than *dataUltimazione* in *lotti* table. |
| | isl_lt_et_ia | Percentage of tuples that meet the following Intrarelational Constraint:<br>- *ImportoSommeLiquidate* must be less than or equal to *ImportoAggiudicazione* in *lotti* table. |
| | lot_has_participant | Percentage of tuples that meet the following Interrelational Constraint:<br>- When a lot has a successful tenderer it must have at least one participant. |
| | successfulTenderer_is_participant | Percentage of tuples that meet the following Interrelational Constraint:<br>- A successful tenderer of a lot must be a participant for that lot. |
| | successfulTenderer_amountPaid | Percentage of tuples that meet the following Interrelational Constraint:<br>- When the successful tenderer is not present for a lot, the amount paid (importoSommeLiquidate) must be zero for that lot. |
| | successfulTenderer_awardAmount | Percentage of tuples that meet the following Interrelational Constraint:<br>- When there is a successful tenderer the award amount (importoAggiudicazione) must be different by zero. |

**Figure 20:** Dimensions for Public Contracts Data

*Chapter 5*

# 5. Evaluation of Public Contracts Data Quality

After checking the state of the art of the models for the quality evaluation and after introducing the framework to be used for public contracts, we apply this framework for the evaluation of the data published by different Italian Universities with the aim of obtaining a ranking of Universities based on quality of the published data.To perform the analysis, and then to query the database and apply the algorithms of calculation of the metrics, we used the Java language.

## 5.1 Data Selection

As we explained in Chapter 2, public administrations are obliged to publish, within January 31th each year, the summary tables of the previous year in the sub section 'Bandi di gara e contratti' of the 'Amministrazione Trasparente' section of its corporate website. To select universities on which to draw up the ranking, we used the overall ranking of the newspaper *'Il Sole 24 Ore'* referring to the year 2014. We have selected the first 25 Universities classified by choosing the year 2014 because the assessment is conducted on the summary tables that are reference to the year 2014. The universities participating in our assessment are shown in Figure 21.

| POSIZIONE | ATENEO |
|---|---|
| 1 | Verona |
| 2 | Trento |
| 3 | Politecnico di Milano |
| 4 | Bologna |
| 5 | Padova |
| 6 | Politecnica delle Marche |
| 7 | Venezia Ca' Foscari |
| 8 | Milano Bicocca |
| 9 | Siena |
| 10 | Politecnico di Torino |
| 11 | Pavia |
| 12 | Piemonte Orientale |
| 13 | Milano Statale |
| 14 | Ferrara |
| 15 | Udine |
| 16 | Macerata |
| 17 | Firenze |
| 18 | Viterbo |
| 19 | Modena e Reggio Emilia |
| 20 | Venezia Iuav |
| 21 | Torino |
| 22 | Roma Foro Italico |
| 23 | Salerno |
| 24 | Pisa |
| 25 | Siena Stranieri |

**Figure 21**: First 25 Universities of the ranking for year 2014 of 'Il Sole 24 ore'

We checked on the websites of the considered universities the presence of the summary tables for the year 2014. The results of research, ended on November 22th, 2015, are shown in Figure 22.

| University | Summary tables |
|---|---|
| Verona | KO |
| Trento | KO |
| Politecnico di Milano | OK |
| Bologna | OK |
| Padova | KO |
| Politecnica delle Marche | OK |
| Venezia Ca' Foscari | OK |

| | |
|---|---|
| Milano Bicocca | OK |
| Siena | KO |
| Politecnico di Torino | OK |
| Pavia | OK |
| Piemonte Orientale | OK |
| Milano Statale | OK |
| Ferrara | OK |
| Udine | KO |
| Macerata | KO |
| Firenze | KO |
| Viterbo | KO |
| Modena e Reggio Emilia | KO |
| Venezia IUAV | KO |
| Torino | OK |
| Roma Foro Italico | KO |
| Salerno | OK |
| Pisa | KO |
| Siena Stranieri | KO |

We can see that on 25 universities only 12 provide summary tables in an XML format while for the other 13 universities we have encountered several problems when searching. The University of Padova and Venezia IUAV provide the summary tables in pdf format while a csv format is used by the University of Trento and University of Udine. In other cases, such as the Universities of Verona, Siena, Macerata, Firenze, Viterbo, Modena e Reggio Emilia, Pisa, Siena Stranieri and Roma Foro Italico the data are not present or there are no data for the year 2014.

## 5.2 Quality Metrics Computation

For the calculation of the metrics chosen to assess the quality of data, it has been taken into account the dimensions described in figure 20. In the following paragraphs it is described each computed metric through the algorithm and the code implemented it is shown to improve the understanding of each of them.

### 5.2.1 Accuracy Computation

To compute the accuracy, the *percentage of cells with correct value(pcvc)* metric is used. The *pcvc* is computed for all the attribute of the *lotti*, *partecipanti* and *aggiudicatari* tables. As was explained in detail in Chapter 3, the data domain is defined in the XML Schema and data outside the domain are handled during transformation stage by assigning them special values. The computation of the

accuracy as the *percentage of cells with correct value* consists on calculating the percentage of cells that do not have these special values.

The algorithm is:

1. For each *entepubblicatore* and for the analysed attribute, checks the values of all the cells and counts the cells with value equal to the special value.
2. For each *entepubblicatore*, calculates the total number of tuples of the table relating to the analysed attribute.
3. Computes the metric:

$$pcvc = 1 - \frac{csfr}{ct}$$

Where:

- csfr: Number of out of domain cells.
- ct: Total number of tuples.

The special values are *NID* for all attributes of text type, the values '0001-01-01' and '3999-12-31' respectively for *dataInizio* and *dataUltimazione* attributes of *lotti* table. To calculate the accuracy of the decimal attributes, the value of the flag associated with them must be checked. Thus, the special values associated to *importoAggiudicazione* and *importoSommeLiquidate*, that must be counted to compute the accuracy, are respectively *flagAgg* equal to 1 and *flagSommeLiq* equal to 1.

We will show in the following the Java code implemented to compute the accuracy of *cig* element. This algorithm is applied to the same way for all the attributes of text type in the tables *lotti*, *partecipanti* and *aggiudicatari*.

```
//Compute the accuracy of 'cig' in the table 'lotti' of 'appalti' database
        public void  accuracy_cig_value(){
                double csfr  = 0;
                double ct = 0;
                double accuracy_cig;
                Number cig;
                String ente = null;
                Statement stm1;
                ResultSet rs1;
                ResultSet rs2;
                try {
                        //call the function to compute the total number of rows in the lotto table
                        rs1 =count_total_number_of_row();
```

```
                        stm1 = conn.createStatement();
                        while(rs1.next()){
                                ct = rs1.getDouble("total_rows");
                                ente = rs1.getString("ente");
                                String query_count_invalid_cig = "SELECT entePubblicatore AS
                                ente,count(*) as invalid_cig
                                FROM appalti.lotti AS l LEFT JOIN appalti.metadati AS m ON
                                l.metadati_urlFile = m.urlFile WHERE CIG= 'NID' AND
                                entePubblicatore = '"+ente+"'";
                                rs2 = stm1.executeQuery(query_count_invalid_cig);
                                while(rs2.next())
                                {
                                        csfr = 0;
                                        csfr = rs2.getDouble("invalid_cig");

                                }
                                //compute the accuracy on lotto
                                accuracy_cig = accuracy_function(csfr, ct);
                                rs2.close();
                        }
                        rs1.close();
                        stm1.close();
                    } catch (SQLException e) {
                        e.printStackTrace();
                    }
                }
//Function used to count the total number of row in the table 'lotto'
        private ResultSet count_total_number_of_row(){
                 ResultSet rs = null;
                 Statement stm;
                String query_count_total_number_of_row = "SELECT entePubblicatore as ente,
                count(*) as total_rows
                 FROM appalti.metadati AS m LEFT JOIN appalti.lotti AS l ON m.urlFile =
                l.metadati_urlFile  GROUP BY entePubblicatore";
                 try {

                        stm = conn.createStatement();
                        rs = stm.executeQuery(query_count_total_number_of_row);
                    } catch (SQLException e) {
                          e.printStackTrace();
                    }
                    return rs;
        }
//Function used to compute the accuracy
        private double accuracy_function(double csfr,double ct){
                    double div;
                    double accuracy_value;

                    div = csfr/ct;
                    accuracy_value = 1-div;

                    return accuracy_value;
                        }
```

The following code shows how to compute the accuracy for the *importoAggiudicazione* attribute. This Algorithm can be used for all the decimal types by indicating the correct flag associated to the attribute.

```
//Compute the accuracy of importoAggiudicazione
        public void accuracy_importoAggiudicazione_value(){

                    double csfr = 0;
```

```
                double ct = 0;
                String ente = null;
                double accuracy_importoAggiudicazione;
                Statement stm1;
                ResultSet rs1,rs2;

                try {
                        rs1 =count_total_number_of_row();
                        stm1 = conn.createStatement();
                    while(rs1.next())
                        {
                          //count the total number of rows on lotto
                          ct = rs1.getDouble("total_rows");
                          ente = rs1.getString("ente");
                          String query_count_invalid_importoAggiudicazione  = "SELECT
                          entePubblicatore,count(*) as invalid_importoAggiudicazione
                          FROM appalti.lotti as l LEFT JOIN appalti.metadati        as m ON
                          l.metadati_urlFile = m.urlFile WHERE flagAgg = '1' AND entePubblicatore=
                          '"+ente+"'";

                          rs2 =
                          stm1.executeQuery(query_count_invalid_importoAggiudicazione);
                          while(rs2.next())
                          {
                            csfr = 0;
                          csfr = rs2.getDouble("invalid_importoAggiudicazione");
                          }
                          accuracy_importoAggiudicazione = accuracy_function(csfr, ct);
                          rs2.close();
                        }
                        rs1.close();
                        stm1.close();
                 } catch (SQLException e) {
                        e.printStackTrace();
                 }

        }
          //Function used to count the total number of row in the table 'lotto'
          private ResultSet count_total_number_of_row(){

              ResultSet rs = null;
              Statement stm;
              String query_count_total_number_of_row = "SELECT entePubblicatore as ente, count(*) as
              total_rows FROM appalti.metadati AS m LEFT JOIN appalti.lotti AS l ON m.urlFile =
              l.metadati_urlFile  GROUP BY entePubblicatore";
              try {
                    stm = conn.createStatement();
                    rs = stm.executeQuery(query_count_total_number_of_row);
              } catch (SQLException e) {
                            e.printStackTrace();
              }
          return rs;
          }
          //Function used to compute the accuracy
          private double accuracy_function(double csfr,double ct){
              double div;
              double accuracy_value;

              div = csfr/ct;
              accuracy_value = 1-div;

            return accuracy_value;
         }
```

### 5.2.2 Completeness Computation

To compute the completeness, the *percentage of complete cells (pcc)* and the *percentage of complete tuples (pcrp)* metrics are computed. To calculate the metrics, the information about the occurrences of the elements specified in the XML Schema, was taken into account and, as explained in chapter 3, in some cases we have assigned special values to the attributes themselves while, in other cases there is the need for a flag, to which a special value is assigned, that help us in the evaluation. The completeness is calculated for all attributes except for *dataInizio* and *dataUltimazione* because they have minimum occurrence equal to zero, that is, they can be not present in a lot but this is not a completeness error.

#### 5.2.2.1 Percentage of Complete Cells

The computation of the *pcc* consists of calculating the percentage of cells that do not have some special values assigned, during the transformation stage, to indicate a completeness error.

The algorithm is:

1. For each *entePubblicatore* and for the analysed attribute, checks the values of all the cells and counts the cells with value equal to the special value.
2. For each *entePubblicatore,* calculates the total number of tuples of the table relating to the analysed attribute.
3. Computes the metric:

$$pcc = 1 - \frac{rm}{rt}$$

Where:

- rm: Number of not complete cells.
- rt: Total number of tuples.

The special values assigned in the case of incompleteness is *null* for text type attribute. In the case of decimal type it is necessary to check the value of the flag associated to the attribute. Thus, to compute the completeness of *importoAggiudicazione* and *importoSommeLiquidate* of *lotti* table, the *flagAgg* and the *flagSommeLiq* must be respectively checked, that is, must be counted the cells

with the *flagAgg* or the *flagSommeLiq* equal to 2. A special case are the *codiceFiscale* and *identificativoFiscaleEstero* in both *partecipanti* and *aggiudicatari* tables, although they are attributes of text type, a flag is associated to them. Since only one of them can be present for one lot, the flag indicates if the attribute is not present because there is the other one or because there is a completeness error. To compute the completeness of these two attributes it is necessary to counts the number of tuples in which *codiceFiscale* (or the identificativoFiscaleEstero) value is *null* and the *codeset* flag is set to 4. For the *ruolo* attribute for both the *partecipanti* and *aggiudicatari*, when the value is *null,* it is necessary to check the flag *codeset*. If the *codeset* is equal to 1 then this must be considered as a completeness error because the participant (or the successful tenderer) is part of group but the *ruolo* element is not present or is blank.

The following will show the Java code to compute the completeness. This Algorithm can be used for any text type attributes in the *lotti*, *partecipanti* and *aggiudicatari* tables.

```
//compute the completeness of cig
   public void completeness_cig_value(){
           double rm = 0;
           double rt = 0;
           String ente = null;
           double completeness_cig;
           Statement stm1;
           ResultSet rs1,rs2;

           try {
           //call the function to compute the total number of rows in the lotto table
            rs1 =count_total_number_of_row();
            stm1 = conn.createStatement();
           while(rs1.next()){
              rt = rs1.getDouble("total_rows");
              ente = rs1.getString("ente");
              String query_count_null_cig = "SELECT entePubblicatore,count(*) as invalid_cig FROM
              appalti.lotti as l LEFT JOIN appalti.metadati as m ON l.metadati_urlFile = m.urlFile WHERE
              cig = 'null' AND entePubblicatore= '"+ente+"'";
              rs2 = stm1.executeQuery(query_count_null_cig);
              while(rs2.next())
              {
                        rm = 0;
                        rm = rs2.getDouble("invalid_cig");

              }
              //call the function to compute the completeness of cig
              completeness_cig = completeness_function(rm, rt);
              rs2.close();
          }
          rs1.close();
          stm1.close();
       } catch (SQLException e) {
                                 e.printStackTrace();
                                  }
   }
//Function used to count the total number of row in the table 'lotti'
```

```
 private ResultSet count_total_number_of_row(){

          ResultSet rs = null;
          Statement stm;
           String query_count_total_number_of_row = "SELECT entePubblicatore   as ente, count(*) as
           total_rows FROM appalti.metadati AS m LEFT JOIN appalti.lotti AS l ON m.urlFile =
           l.metadati_urlFile  GROUP BY entePubblicatore";

          try {
                    stm = conn.createStatement();
               rs = stm.executeQuery(query_count_total_number_of_row);
          } catch (SQLException e) {
            e.printStackTrace();
                                  }
          return rs;
}

//function used to compute the completeness
 private double completeness_function(double rm, double rt){
           double div;
           double completeness_value;
           div = rm/rt;
   completeness_value = 1-(div);

    return completeness_value;
 }
```

The code below shows how to compute the completeness for the *codiceFiscale* attribute in the *partecipanti* table but the same algorithm is applied to compute the completeness of the *codiceFiscale* in the *aggiudicatari* table.

```
//Compute the completeness of 'codiceFiscale' in table 'partecipanti'
   public void completeness_codiceFiscale_value(){
              double rm = 0;
              double rt = 0;
              String ente = null;
              double completeness_codiceFiscale;
              Statement stm;
              ResultSet rs1,rs2;


                   try {
                             //call the function to compute the total number of rows
                      rs1 = count_total_number_of_row();
                             stm = conn.createStatement();
                             int i = 1;
                             while(rs1.next()){
                                rt = rs1.getDouble("total_rows");
                                ente = rs1.getString("ente");
                               //Count the number of rows where codiceFiscale = null
                               // and codeSet = 4
                               String query_count_null_codiceFiscale = "SELECT
                               entePubblicatore,count(*) AS invalid_codiceFiscale
                               FROM (`appalti`.`partecipanti` as p LEFT JOIN appalti.lotti_partecipanti as
                               l_p on p.idPartecipante = l_p.partecipante_idPartecipante) LEFT JOIN
                               appalti.lotti AS l ON l_p.lotto_idCig = l.idCig LEFT JOIN appalti.metadati
                               AS m ON l.metadati_urlFile = m.urlFile WHERE codiceFiscale='null' AND
                               codeset = '4' AND entepubblicatore='"+ente+"'";
                               rs2 = stm.executeQuery(query_count_null_codiceFiscale);
                               while(rs2.next())
                                {
                                         rm = 0;
                                         rm = rs2.getDouble("invalid_codiceFiscale");
                                 }
                               completeness_codiceFiscale = completeness_function(rm, rt);
```

```
                                rs2.close();
                            }
                            rs1.close();
                            stm.close();
                        } catch (SQLException e) {
                            e.printStackTrace();
                        }

                }

    //Function used to compute the total number of rows
    private ResultSet count_total_number_of_row(){
            Statement stm;
            ResultSet rs = null;

             String count_number_of_row = "SELECT count(distinct(idPartecipante)) as total_rows,
             entePubblicatore as ente FROM appalti.metadati as m LEFT JOIN appalti.lotti as l on
             m.urlFile=l.metadati_urlFile LEFT JOIN appalti.lotti_partecipanti as l_p on l.idCig =
             l_p.lotto_idCig LEFT JOIN appalti.partecipanti as p ON l_p.partecipante_idPartecipante =
             p.idPartecipante GROUP BY entePubblicatore";
             try {
                 stm = conn.createStatement();
                      rs = stm.executeQuery(count_number_of_row);

                  } catch (SQLException e) {
                            e.printStackTrace();
                                    }
            return rs;
            }
            //Function to compute the completeness
            private double completeness_function(double rm, double rt){
                        double div;
                        double completeness_value;
                        div = rm/rt;
                        completeness_value = 1-(div);

                        return completeness_value;
            }
```

### 5.2.2.2 Percentage of Complete tuples

The *percentage of complete tuples (pcrp)* computes how many tuples have all the attributes and gives an idea of how missing values are distributed in the analysed table.

The algorithm is:

1. For the analysed table and for each *entePubblicatore*, checks all the values of the tuple and counts the tuples that have at least one missing value.
2. Calculates the total number of tuples for the specific *entePubblicatore*.
3. Computes the metric:

$$pcrp = 1 - \frac{rm}{rt}$$

Where:

- rm: Number of tuples with at least one missing value.
- rt: Total number of tuples.

In the following code is shown how is computed the *pcrp* metric for the *lotti* table. The same algorithm is used to calculate the metric for *partecipanti* and *aggiudicatari* tables.

```
//Compute the completeness on the row of the table
//Count the rows that have at least one 'null' value
  public void completeness_on_lotto(){
            double rm = 0;
            double rt = 0;
            String ente = null;
            double completeness_on_lotto;
            Statement stm1;
            ResultSet rs1,rs2;
            Number rowComp;

            try {
                 rs1 =count_total_number_of_row();
                   stm1 = conn.createStatement();
                   while(rs1.next())
                   {
                        rt = rs1.getDouble("total_rows");
                       ente = rs1.getString("ente");
                        String count_null_value =  "SELECT entePubblicatore,count(*) as invalid_row
                        FROM appalti.lotti as l LEFT JOIN appalti.metadati as m ON l.metadati_urlFile =
                        m.urlFile WHERE (cig='null' or codiceFiscaleProp='null' or denominazione ='null' or
                        oggetto = 'null' or sceltaContraente = 'null' or flagAgg = '2' or flagSommeLiq = '2')
                        AND entePubblicatore= '"+ente+"'";
                         rs2 = stm1.executeQuery(count_null_value);
                         while(rs2.next())
                         {
                                  rm = 0;
                                  rm = rs2.getDouble("invalid_row");
                          }
                         completeness_on_lotto = completeness_function(rm, rt);
                         rs2.close();
                        }
                      rs1.close();
                      stm1.close();
            } catch (SQLException e) {
                      e.printStackTrace();
            }

 }
//Function used to count the total number of row in the table 'lotto'
  private ResultSet count_total_number_of_row(){

            ResultSet rs = null;
            Statement stm;
           String query_count_total_number_of_row = "SELECT entePubblicatore as ente, count(*) as
           total_rows FROM appalti.metadati AS m LEFT JOIN appalti.lotti AS l ON m.urlFile =
           l.metadati_urlFile  GROUP BY entePubblicatore";

            try {
                 stm = conn.createStatement();
                 rs = stm.executeQuery(query_count_total_number_of_row);
            } catch (SQLException e) {
                            e.printStackTrace();
                                    }
            return rs;
```

```
}

  //function used to compute the completeness
   private double completeness_function(double rm, double rt){
              double div;
              double completeness_value;
               div = rm/rt;
               completeness_value = 1-(div);

      return completeness_value;
   }
```

### 5.2.3 Duplication

To compute the duplication, the *dup_participant* and the *dup_tenderer* metrics are used. The duplication is computed only for the participant and the successful tenderers, that is, we guess that the same participant, as well as the same successful tenderer, cannot appear more than once in a lot. Thus, the *dup_participant* metric consists on counting the participant duplicated while, the dup_tenderer counts the duplicated successful tenderers.

The algorithm is:

1. For each lot of the same *entePubblicatore*, counts the participant that are duplicated.
2. Counts, the total number of duplicated participant, for the same *entePubblicatore,* adding up the number of duplicates found on each lot.

The *lotti_partecipanti* and the *lotti_aggiudicatari* table keep track, respectively, of the participants taking part in the lots and successful tenderers who win lots. This two tables are used to compute the duplication metrics.

The following will show the code implemented to compute the total number of duplicated participant. The same code can be used for the successful tenderers by changing the tables and the attributes on which the query is executed.

```
    public void duplication_partecipanti()
    {
          Statement stm;
          ResultSet rs;
          double dup = 0;
          String ente = null;
          String query_count_duplicates = "SELECT entePubblicatore AS ente ,SUM(duplicates) as
          duplicates FROM (SELECT entePubblicatore,
          lotto_idCig,l_p.partecipante_idPartecipante,(count(*)-1) as duplicates FROM
          (appalti.lotti_partecipanti AS l_p LEFT JOIN appalti.lotti AS l ON l_p.lotto_idCig = l.idCig) LEFT
          JOIN appalti.metadati AS m ON l.metadati_urlFile = m.urlFile GROUP BY  entePubblicatore,
          lotto_idCig, l_p.partecipante_idPartecipante HAVING count(*)>1) AS A GROUP BY
          entePubblicatore";
```

```
        try {
             stm = conn.createStatement();
             rs = stm.executeQuery(query_count_duplicates);
              while(rs.next())
              {
                    ente = rs.getString("ente");
                    dup = rs.getDouble("duplicates");
             }
             rs.close();
           stm.close();
        } catch (SQLException e) {
                              e.printStackTrace();
                    }

  }
```

### 5.2.4 Consistency

The consistency is assessed through the definition of certain integrity constraints. For each of these constraints a metric is defined.

The intrarelational constraints are computed with the following metrics:

- *cf_ife_partecipanti*
- *cf_ife_aggiudicatari*
- *cf_eq_zero_partecipanti*
- *cf_eq_zero_aggiudicatari*
- *check_on_date*
- *isl_lt_et_ia*

while for the interrelational constraints the metrics computed are:

- lot_has_participant
- successfulTenderer_isparticipant
- successfulTenderer_amountPaid
- successfulTenderer_awardAmount

#### 5.2.4.1 cf_ife_partecipanti

The computation of this metrics consists on calculating the percentage of tuples that meet the defined integrity constraint, that is, the total number of tuples in the *partecipanti* table in which both *codiceFiscale* and the *identificativoFiscaleEstero* are not simultaneously present.

The algorithm is:

1. For each *entePubblicatore,* checks the value of *codeset* attribute in *partecipanti* table and counts the cells that do not meet the constraint.
2. For each *entePubblicatore*, counts the total number of tuples in the *partecipanti* table.
3. Compute the metric:

$$cf\_ife\_partecipanti = 1 - \frac{cm}{ct}$$

where:

- *cm:* number of cells of *partecipanti* table where *codeset* = 3.
- *ct:* total number of tuples of *partecipanti* table.

As we already mentioned in chapter 3, the flag *codeset* is defined in both the *partecipanti* and *aggiudicatari* tables and its value depend on the presence or absence of the *codiceFiscale* and *identificativoFiscaleEstero*. If both the attributes are *not null,* the codeset is set to 3 and this codeset help us in the evaluation of the consistency.

The following code shows the implementation of the metric.

```
public void cfd_ife_partecipanti_constraint()
        {
            Statement stm;
            ResultSet rs1,rs2;
            String ente = null;
            double cm = 0;
           double ct = 0;
            double consistency_of_codeset_partecipanti;

             try {
                    //call the function to compute the total number of rows
                    rs1 = count_total_number_of_row_partecipanti();
                    stm = conn.createStatement();
                     while(rs1.next())
                     {
                            ct = rs1.getDouble("total_rows");
                            ente = rs1.getString("ente");
                             String query_count_invalid_participant = "SELECT entepubblicatore,
                             COUNT(*) AS invalid_participant FROM (appalti.partecipanti AS p LEFT JOIN
                             appalti.lotti_partecipanti AS l_p ON p.idPartecipante =
                             partecipante_idPartecipante) LEFT JOIN appalti.lotti AS l ON l_p.lotto_idCig =
                             l.idCig LEFT JOIN appalti.metadati AS m ON l.metadati_urlFile = m.urlFile
```

```
                        WHERE codeset  = '3' and entePubblicatore = '"+ente+"'";
                        rs2 = stm.executeQuery(query_count_invalid_participant);
                        while(rs2.next())
                          {
                                cm = 0;
                                cm = rs2.getDouble("invalid_participant");
                          }
                        consistency_of_codeset_partecipanti = consistency_function(cm, ct);
                        rs2.close();
                    }
                rs1.close();
                stm.close();
                } catch (SQLException e) {
                        e.printStackTrace();
                }
  }
//function used to count the total number of row of 'partecipanti' table
  private ResultSet count_total_number_of_row_partecipanti(){
            Statement stm;
            ResultSet rs = null;

                String count_number_of_row = "SELECT count(distinct(idPartecipante)) as total_rows,
                entePubblicatore as ente FROM appalti.metadati as m LEFT JOIN appalti.lotti as l on
                m.urlFile=l.metadati_urlFile LEFT JOIN appalti.lotti_partecipanti as l_p on l.idCig =
                l_p.lotto_idCig LEFT JOIN appalti.partecipanti as p ON l_p.partecipante_idPartecipante
                = p.idPartecipante GROUP BY entePubblicatore";
            try {
                    stm = conn.createStatement();
                    rs = stm.executeQuery(count_number_of_row);
        } catch (SQLException e) {
                        e.printStackTrace();
             }
                return rs;
          }
//function used to compute the number of row wich does not violate the constraints
  private double consistency_function(double cm, double ct){
                    double div;
                    double completeness_value;
                    div = cm/ct;
                completeness_value = 1-(div);

             return completeness_value;
  }
```

#### 5.2.4.2 cf_ife_aggiudicatari

This metric is similar to the *cf_ife_partecipanti* explained above but it is computed on the *aggiudicatari* table instead of the *partecipanti* table. Thus, the *cf_ife_aggiudicatari* consists on calculating the total number of tuples in the *aggiudicatari* table in which both *codiceFiscale* and the *identificativoFiscaleEstero* are not simultaneously present.

The algorithm is:

1. For each *entePubblicatore,* checks the value of *codeset* attribute in *aggiudicatari* table and counts the cells that do not meet the constraint.
2. For each *entePubblicatore*, counts the total number of tuples in the *aggiudicatari* table.

3. Compute the metric:

$$cf\_ife\_aggiudicatari = 1 - \frac{cm}{ct}$$

where:

- *cm:* number of cells of *aggiudicatari* table where *codeset* = 3.
- *ct:* total number of tuples of *aggiudicatari* table.

The code below shows the implementation of the *cf_ife_aggiudicatari metric.*

```
//evalutation of the following integrity constraint:
//For an aggiudicatario  must appear one and only one among <codiceFiscale>
<identificativoFiscaleEstero>.  They cannot cannot appear together
  public void cdf_ide_aggiudicatari_constraint()
  {
            Statement stm;
            ResultSet rs1,rs2;
            String ente = null;
            double cm = 0;
            double ct = 0;
            double consistency_of_codeset_aggiudicatari;
            Number cfd_ife_agg;

            try {
                //call the function to compute the total number of rows on aggiudicatari
               rs1 = count_total_number_of_row_aggiudicatari();
               stm = conn.createStatement();
                while(rs1.next()){
                     ct = rs1.getDouble("total_rows");
                    ente = rs1.getString("ente");
                    String query_count_invalid_aggiudicatari = "SELECT entepubblicatore, COUNT(*) as
                    invalid_aggiudicatari FROM(appalti.aggiudicatari AS a LEFT JOIN
                    appalti.lotti_aggiudicatari as l_a ON a.idAggiudicatario =
                    l_a.aggiudicatari_idAggiudicatario) LEFT JOIN appalti.lotti AS l ON l_a.lotto_idCig =
                    l.idCig  LEFT JOIN appalti.metadati AS m on l.metadati_urlFile = m.urlFile WHERE
                    a.codeset = '3' AND entePubblicatore = '"+ente+"'";
                    rs2 = stm.executeQuery(query_count_invalid_aggiudicatari);
                    while(rs2.next())
                    {
                        cm = 0;
                        cm = rs2.getDouble("invalid_aggiudicatari");
                    }
                    consistency_of_codeset_aggiudicatari = consistency_function(cm, ct);
                    rs2.close();
                }
                rs1.close();
                stm.close();
                    } catch (SQLException e) {
                                e.printStackTrace();
                    }

          }
//function used to count the total number of row of 'aggiudicatari' table
  private ResultSet count_total_number_of_row_aggiudicatari(){
```

```
            Statement stm;
            ResultSet rs = null;

             String count_number_of_row = "SELECT count(distinct(idAggiudicatario)) AS total_rows,
             entePubblicatore AS ente FROM appalti.metadati AS m LEFT JOIN appalti.lotti AS l ON
             m.urlFile = l.metadati_urlFile LEFT JOIN appalti.lotti_aggiudicatari AS l_a ON l.idCig =
             l_a.lotto_idCig LEFT JOIN appalti.aggiudicatari AS a ON l_a.aggiudicatari_idAggiudicatario =
             a.idAggiudicatario GROUP BY entePubblicatore";
             try {
                     stm = conn.createStatement();
                     rs = stm.executeQuery(count_number_of_row);

              } catch (SQLException e) {
                             e.printStackTrace();
                     }
                     return rs;
           }
//function used to compute the number of row wich does not violate the constraints
  private double consistency_function(double cm, double ct){
              double div;
             double completeness_value;

              div = cm/ct;
             completeness_value = 1-(div);

             return completeness_value;
            }
}
```

#### 5.2.4.3 cf_eq_zero_partecipanti

The metric cf_eq_zero_partecipanti consists of calculating the percentage of tuple with correct value, that is, the tuples that meet the constraint that *codiceFiscale* attribute in the *partecipanti* table must be not equal neither to *00000000000* nor to *0000000000000000*.

The algorithm is:

1. For each *entePubblicatore,* checks the value of *codiceFiscale* attribute in *partecipanti* table and counts the cells that do not meet the constraint.
2. For each *entePubblicatore*, counts the total number of tuples in the *partecipanti* table.
3. Compute the metric:

$$cf\_eq\_zero\_partecipanti = 1 - \frac{cm}{ct}$$

where:

- *cm:* number of cells of *partecipanti* table where *codiceFiscale* = *000000000000* or *codiceFiscale* = *0000000000000000*.
- *ct:* total number of tuples of *partecipanti* table.

The code shows how the metric is implemented.

```
//Evaluation of the following integrity constraint
//CodiceFiscale must be not equal to 00000000000 or 0000000000000000 in partecipanti table
  public void cdf_equal_zero_partecipanti()
  {
          Statement stm;
          ResultSet rs1,rs2;
          String ente = null;
          double cm = 0;
          double ct = 0;
          double consistency_of_cdf_partecipante;

          try {
             //call the function to compute the total number of rows
             rs1 = count_total_number_of_row_partecipanti();
              stm = conn.createStatement();
              while(rs1.next())
              {
                    ct = rs1.getDouble("total_rows");
                    ente = rs1.getString("ente");
                   String query_invalid_codiceFiscale_partecipanti = "SELECT entePubblicatore,
                   COUNT(*) AS invalid_codiceFiscale FROM (appalti.partecipanti AS p LEFT JOIN
                   appalti.lotti_partecipanti AS l_p ON p.idPartecipante = partecipante_idPartecipante)
                   LEFT JOIN appalti.lotti AS l ON l_p.lotto_idCig = l.idCig LEFT JOIN appalti.metadati AS
                   m ON l.metadati_urlFile = m.urlFile WHERE (codiceFiscale = '00000000000' OR
                   codiceFiscale= '0000000000000000') AND entePubblicatore = '"+ente+"'";
                   rs2 = stm.executeQuery(query_invalid_codiceFiscale_partecipanti);
                   while(rs2.next())
                   {
                       cm = 0;
                       cm = rs2.getDouble("invalid_codiceFiscale");
                     }
                    consistency_of_cdf_partecipante = consistency_function(cm, ct);
                        }
                     } catch (SQLException e) {
                                e.printStackTrace();
                     }
        }
//function used to count the total number of row of 'partecipanti' table
  private ResultSet count_total_number_of_row_partecipanti(){
               Statement stm;
               ResultSet rs = null;

                   String count_number_of_row = "SELECT count(distinct(idPartecipante)) as total_rows,
                   entePubblicatore as ente FROM appalti.metadati as m LEFT JOIN appalti.lotti as l on
                   m.urlFile=l.metadati_urlFile LEFT JOIN appalti.lotti_partecipanti as l_p on l.idCig =
                   l_p.lotto_idCig LEFT JOIN appalti.partecipanti as p ON l_p.partecipante_idPartecipante
                   = p.idPartecipante GROUP BY entePubblicatore";
               try {
                        stm = conn.createStatement();
                        rs = stm.executeQuery(count_number_of_row);
         } catch (SQLException e) {
                                e.printStackTrace();
                }
                      return rs;
           }
//function used to compute the number of row wich does not violate the constraints
  private double consistency_function(double cm, double ct){
                        double div;
                        double completeness_value;
                        div = cm/ct;
                  completeness_value = 1-(div);

                return completeness_value;
  }
```

#### 5.2.4.4 cf_eq_zero_aggiudicatari

This metric is similar to the *cf_eq_zero_partecipanti* but it computes the percentage of cells that meet the constraint considering the *aggiudicatari* table.

The algorithm is:

1. For each *entePubblicatore,* checks the value of *codiceFiscale* attribute in *aggiudicatari* table and counts the cells that do not meet the constraint.
2. For each *entePubblicatore*, counts the total number of tuples in the *aggiudicatari* table.
3. Compute the metric:

$$cf\_eq\_zero\_aggiudicatari = 1 - \frac{cm}{ct}$$

where:

- *cm:* number of cells of *aggiudicatari* table where *codiceFiscale = 000000000000* or *codiceFiscale = 0000000000000000*.
- *ct:* total number of tuples of *aggiudicatari* table.

The following code shows how the metric is implemented.

```
//Evaluation of the following integrity constraint
//CodiceFiscale must be not equal to 00000000000 or 0000000000000000 in aggiudicatari table
  public void cdf_equal_zero_aggiudicatari()
  {
            Statement stm;
            ResultSet rs1,rs2;
            String ente = null;
            double cm = 0;
            double ct = 0;
            double consistency_of_cdf_aggiudicatario;

      try {
//call the function to compute the total number of rows on aggiudicatari
                        rs1 = count_total_number_of_row_aggiudicatari();
                        stm = conn.createStatement();
          while(rs1.next())
                        {
                            ct = rs1.getDouble("total_rows");
                            ente = rs1.getString("ente");
                                String query_invalid_codiceFiscale_aggiudicatari = "SELECT
                                entepubblicatore, COUNT(*) AS invalid_codiceFiscale
                                FROM(appalti.aggiudicatari AS a LEFT JOIN appalti.lotti_aggiudicatari as l_a
                                ON a.idAggiudicatario = l_a.aggiudicatari_idAggiudicatario) LEFT JOIN
                                appalti.lotti AS l ON l_a.lotto_idCig = l.idCig  LEFT JOIN appalti.metadati AS m
                                on l.metadati_urlFile = m.urlFile WHERE (codiceFiscale = '00000000000' OR
                                codiceFiscale = '0000000000000000') AND entePubblicatore = '"+ente+"'";
                            rs2 = stm.executeQuery(query_invalid_codiceFiscale_aggiudicatari);
                           while(rs2.next())
                           {
```

```
                                cm = 0;
                                cm = rs2.getDouble("invalid_codiceFiscale");
                            }
                              consistency_of_cdf_aggiudicatario =   consistency_function(cm, ct);
                              rs2.close();
                        }
                        rs1.close();
                        stm.close();
                        } catch (SQLException e) {
                                e.printStackTrace();
                        }
}
//function used to count the total number of row of 'aggiudicatari' table
  private ResultSet count_total_number_of_row_aggiudicatari(){
                Statement stm;
                ResultSet rs = null;

                String count_number_of_row = "SELECT count(distinct(idAggiudicatario)) AS total_rows,
                entePubblicatore AS ente FROM appalti.metadati AS m LEFT JOIN appalti.lotti AS l ON
                m.urlFile = l.metadati_urlFile LEFT JOIN appalti.lotti_aggiudicatari AS l_a ON l.idCig =
                l_a.lotto_idCig LEFT JOIN appalti.aggiudicatari AS a ON l_a.aggiudicatari_idAggiudicatario
                = a.idAggiudicatario GROUP BY entePubblicatore";
                try {
                        stm = conn.createStatement();
                        rs = stm.executeQuery(count_number_of_row);
                  } catch (SQLException e) {
                                e.printStackTrace();
                   }
                     return rs;
   }
//function used to compute the number of row wich does not violate the constraints
  private double consistency_function(double cm, double ct){
                double div;
               double completeness_value;

                div = cm/ct;
               completeness_value = 1-(div);

                return completeness_value;
             }
```

#### 5.2.4.5 check_on_date

The metric computes the percentage of tuples that meet the constraint, that is, the tuples of the *lotti* table for which *dataInizio* is less recent than *dataUltimazione*.

The algorithm is:

1. For each *entePubblicatore,* checks the value of *dataInizio* and *dataUltimazione* attributes of *lotti* table and counts the number of tuples that do not meet the constraint.
2. For each *entePubblicatore*, counts the total number of tuples in the *lotti* table.
3. Compute the metric:

$$check\_on\_date = 1 - \frac{cm}{ct}$$

where:

- *cm:* number of cells of *lotti* table where *dataInizio* is more recent than *dataUltimazione.*
- *ct:* total number of tuples of *lotti* table.

The following code shows how the metric is implemented.

```
//evaluates the integrity constraint
//dataInizio MUST be before dataUltimazione
  public void check_on_date()
        {
            Statement stm;
               ResultSet rs1,rs2;
               String ente =null;
               double cm = 0;
               double ct = 0;
               double invalid_data = 0;

               try {
                  //call the function to compute the total number of rows on aggiudicatari
                  rs1 = count_total_number_of_row_lotto();
                  stm = conn.createStatement();
                  while(rs1.next())
                   {
                        ct = rs1.getDouble("total_rows");
                        ente = rs1.getString("ente");
                        String query_invalid_data = "SELECT entePubblicatore as ente, count(*) AS
                        invalid_data FROM appalti.lotti AS l LEFT JOIN appalti.metadati AS m ON
                        l.metadati_urlFile = m.urlFile WHERE dataInizio > dataUltimazione and
                        entePubblicatore = '"+ente+"'";
                         rs2 = stm.executeQuery(query_invalid_data);
                         while(rs2.next())
                         {
                               cm = 0;
                               cm = rs2.getDouble("invalid_data");
                         }
                        invalid_data = consistency_function(cm, ct);
                        rs2.close();
                        }
                     rs1.close();
                     stm.close();
                  } catch (SQLException e) {
                            e.printStackTrace();
                  }
        }
//function used to count the total number of row of the 'lotto' table
  private ResultSet count_total_number_of_row_lotto(){

                ResultSet rs = null;
                Statement stm;

               String query_count_total_number_of_row = "SELECT entePubblicatore as ente, count(*)
               as total_rows FROM appalti.metadati AS m LEFT JOIN appalti.lotti AS l ON m.urlFile =
               l.metadati_urlFile  GROUP BY entePubblicatore";
```

```
                try {
                        stm = conn.createStatement();
                        rs = stm.executeQuery(query_count_total_number_of_row);
                    } catch (SQLException e) {
                                e.printStackTrace();
                            }
                return rs;
            }
//function used to compute the number of row wich does not violate the constraints
  private double consistency_function(double cm, double ct){
                double div;
                double completeness_value;

                 div = cm/ct;
                 completeness_value = 1-(div);

              return completeness_value;
           }
```

#### 5.2.4.6 isl_lt_et_ia

The isl_lt_et_ia calculates the percentage of tuples that meet the constraint, that is, the tuples for which the *importoSommeLiquidate* value is less than or equal to the *importoAggiudicazione* value.

The algorithm is:

1. For each *entePubblicatore,* checks the value of *importoAggiudicazione* and *importoSommeLiquidate* attributes of *lotti* table and counts the number of tuples that do not meet the constraint.
2. For each *entePubblicatore*, counts the total number of tuples in the *lotti* table.
3. Compute the metric:

$$isl\_lt\_et\_ia = 1 - \frac{cm}{ct}$$

where:

- *cm:* number of cells of *lotti* table where *importoSommeLiquidate* is greater than *importoAggiudicazione.*
- *ct:* total number of tuples of *lotti* table.

The following code shows how the metric is implemented.

```
//evaluates the following integrity constraint
//The importoSommeLiquidate MUST be less than or equal to the ImportoAggiudicazione
public void isl_lt_et_ia()
```

```
        {
            Statement stm;
            ResultSet rs1,rs2;
            String ente = null;
            double cm = 0 ;
            double ct;
            double ISL_Lt_Et_IA;

             try {
                  //call the function to compute the total number of rows on aggiudicatari
                  rs1 = count_total_number_of_row_lotto();
                  stm = conn.createStatement();
                  while(rs1.next())
                    {
                       ct = rs1.getDouble("total_rows");
                       ente = rs1.getString("ente");
                       String query_invalid_import = "SELECT entePubblicatore as ente, count(*) as
                       invalid_import FROM appalti.lotti AS l LEFT JOIN appalti.metadati AS m ON
                       l.metadati_urlFile = m.urlFile  WHERE importoSommeLiquidate >
                       importoAggiudicazione and entePubblicatore = '"+ente+"'";
                       rs2 = stm.executeQuery(query_invalid_import);
                       while(rs2.next())
                       {
                            cm = 0;
                            cm = rs2.getDouble("invalid_import");
                       }
                       ISL_Lt_Et_IA = consistency_function(cm, ct);
                       rs2.close();
                    }
                    rs1.close();
                    stm.close();
                  } catch (SQLException e) {
                            e.printStackTrace();
                  }
  }
//function used to count the total number of row of the 'lotto' table
  private ResultSet count_total_number_of_row_lotto(){

              ResultSet rs = null;
              Statement stm;

               String query_count_total_number_of_row = "SELECT entePubblicatore as ente, count(*)
               as total_rows FROM appalti.metadati AS m LEFT JOIN appalti.lotti AS l ON m.urlFile =
               l.metadati_urlFile  GROUP BY entePubblicatore";

               try {
                        stm = conn.createStatement();
                        rs = stm.executeQuery(query_count_total_number_of_row);
                     } catch (SQLException e) {
                                e.printStackTrace();
                                }
                     return rs;
          }
//function used to compute the number of row wich does not violate the constraints
  private double consistency_function(double cm, double ct){
                double div;
                double completeness_value;

                div = cm/ct;
               completeness_value = 1-(div);

              return completeness_value;
           }
```

#### 5.2.4.7 lot_has_participant

The lot_has_participant metrics compute the percentage of lots the meet the interrelational constraint that states that if a lot has a successful tenderer it must have at least one participant.

The algorithm is:

1. For each *entePubblicatore,* counts the total number of tuples that do not meet the constraint.
2. For each *entePubblicatore*, counts the total number of tuples in the *lotti* table.
3. Compute the metric:

$$lot\_has\_participant = 1 - \frac{rm}{rt}$$

where:

- *rm:* number of lots that have a successful tenderer but do not have participants.
- *rt:* total number of tuples of *lotti* table.

The following code shows how the metric is implemented.

```
//evaluation of the following constraint
//if there is an aggiudicatario there MUST be at least one participant
  public void lot_has_participant_constraint()
  {
            Statement stm;
            ResultSet rs1,rs2;
            String ente = null;
            double rm = 0;
            double rt;
            double invalid_rows;
            Label lente;
            Number part_const;

            try {
                        //call the function to compute the total number of rows on aggiudicatari
                        rs1 = count_total_number_of_row_lotto();
                        stm = conn.createStatement();
                        int i = 1;
                        while(rs1.next())
                        {

                                    rt = rs1.getDouble("total_rows");
                                    ente = rs1.getString("ente");
                                    lente = new Label(0,i,ente);
```

```
                                try {
                                        sheet10.addCell(lente);
                                } catch (RowsExceededException e) {
                                        e.printStackTrace();
                                } catch (WriteException e) {
                                        e.printStackTrace();
                                }
                                String query = "SELECT
                                entePubblicatore,count(DISTINCT(l_a.lotto_idCig)) AS
                                invalid_aggiudicatari FROM(appalti.lotti_aggiudicatari AS l_a LEFT JOIN
                                appalti.lotti AS l ON l_a.lotto_idCig = l.idCig) LEFT JOIN
                                appalti.lotti_partecipanti AS l_p ON l.idCig = l_p.lotto_idCig LEFT JOIN
                                appalti.metadati AS m ON l.metadati_urlFile = m.urlFile WHERE
                                l_p.partecipante_idPartecipante is null AND entePubblicatore = '"+ente+"'";
                                rs2 = stm.executeQuery(query);
                                while(rs2.next())
                                {
                                        rm = 0;
                                        rm = rs2.getDouble("invalid_aggiudicatari");

                                }
                                invalid_rows = consistency_function(rm, rt);
                                 rs2.close();
                        }
                        rs1.close();
                        stm.close();
                } catch (SQLException e) {
                        e.printStackTrace();
        }
 }
//function used to count the total number of row of the 'lotto' table
  private ResultSet count_total_number_of_row_lotto(){

                 ResultSet rs = null;
                Statement stm;

                String query_count_total_number_of_row = "SELECT entePubblicatore as ente, count(*)
                as total_rows FROM appalti.metadati AS m LEFT JOIN appalti.lotti AS l ON m.urlFile =
                l.metadati_urlFile  GROUP BY entePubblicatore";

                try {
                        stm = conn.createStatement();
                        rs = stm.executeQuery(query_count_total_number_of_row);
                    } catch (SQLException e) {
                                e.printStackTrace();
                            }
                    return rs;
        }
//function used to compute the number of row wich does not violate the constraints
  private double consistency_function(double rm, double rt)
  {
                    double div;
                    double completeness_value;

                    div = rm/rt;
                    completeness_value = 1-(div);

                   return completeness_value;
 }
```

#### 5.2.4.8 successfulTenderer_isparticipant

The defined constraint states that if a lot has a successful tenderer this must be a participant of that lot. The metric computes the percentage of tuples that meet this constraint.

The algorithm is:

1. For each *entePubblicatore,* counts the total number of tuples that do not meet the constraint.
2. For each *entePubblicatore*, counts the total number of tuples in the *lotti* table.
3. Compute the metric:

$$successfulTenderer\_isparticipant = 1 - \frac{rm}{rt}$$

where:

- *rm:* number of lots that have a successful tenderer but this is not a participant.
- *rt:* total number of tuples of *lotti* table.

The following code shows how the metric is implemented.

```
public void successfulTenderer_isparticipant_constraint()
  {
            Statement stm;
            ResultSet rs1,rs2;
            String ente = null;
            double rm = 0;
            double rt;
            double invalid_rows;

            try {
                  //call the function to compute the total number of rows on aggiudicatari
                    rs1 = count_total_number_of_row_lotto();
                    stm = conn.createStatement();
                    int i = 1;
                    while(rs1.next())
                    {
                              rt = rs1.getDouble("total_rows");
                              ente = rs1.getString("ente");
                             String query = "SELECT count(DISTINCT(l_a.lotto_idCig)) as
                             invalid_aggiudicatario FROM (appalti.lotti_aggiudicatari AS l_a  LEFT JOIN
                             appalti.aggiudicatari AS a ON l_a.aggiudicatari_idAggiudicatario =
                             a.idAggiudicatario) LEFT JOIN appalti.lotti AS l ON l_a.lotto_idCig = l.idCig
                             LEFT JOIN appalti.metadati AS m ON l.metadati_urlFile = m.urlFile LEFT
                             JOIN  (appalti.lotti_partecipanti AS l_p LEFT JOIN appalti.partecipanti as p
```

```
                                ON l_p.partecipante_idPartecipante = p.idPartecipante) ON  l.idCig =
                                l_p.lotto_idCig AND a.codiceFiscale = p.codiceFiscale AND
                                a.identificativoFiscaleEstero = p.identificativoFiscaleEstero WHERE
                                p.codiceFiscale is null AND entePubblicatore= '"+ente+"'";
                                 rs2 = stm.executeQuery(query);
                                 while(rs2.next())
                                 {
                                        rm = 0;
                                        rm = rs2.getDouble("invalid_aggiudicatario");
                                 }
                                rows = consistency_function(rm, rt);
                                rs2.close();
                                agg_const = new Number(2,i,invalid_rows);
                        }
                        rs1.close();

                        stm.close();
                } catch (SQLException e) {
                        e.printStackTrace();
                }
        }
//function used to count the total number of row of the 'lotto' table
  private ResultSet count_total_number_of_row_lotto(){

                 ResultSet rs = null;
                Statement stm;

               String query_count_total_number_of_row = "SELECT entePubblicatore as ente, count(*)
               as total_rows FROM appalti.metadati AS m LEFT JOIN appalti.lotti AS l ON m.urlFile =
               l.metadati_urlFile  GROUP BY entePubblicatore";

                try {
                        stm = conn.createStatement();
                        rs = stm.executeQuery(query_count_total_number_of_row);
                    } catch (SQLException e) {
                                e.printStackTrace();
                            }
                    return rs;
          }
//function used to compute the number of row wich does not violate the constraints
  private double consistency_function(double rm, double rt)
  {
                    double div;
                    double completeness_value;

                    div = rm/rt;
                    completeness_value = 1-(div);

                   return completeness_value;
 }
```

### 5.2.4.9 successfulTenderer_amountPaid

This constraint imply that if a lot does not have a successful tenderer the amount paid must be zero. The relative metric calculates the percentage of tuples that meet the constraint.

The algorithm is:

1. For each *entePubblicatore,* counts the number of tuples that do not meet the constraint.

2. For each *entePubblicatore*, counts the total number of tuples in the *lotti* table.
3. Compute the metric:

$$successfulTenderer\_amoutPaid = 1 - \frac{rm}{rt}$$

where:

- *rm:* number of lots that have *importoSommeLiquidate* attribute value different by zero but do not have any successful tenderer.
- *ct:* total number of tuples of *lotti* table.

In the following code is shown the implementation of the metric.

```
//evaluaton of the following constraint
 //when the aggiudicatario is not present the importoSommeLiquidate MUST be zero
  public void successfulTender_amountPaid_constraint()
  {
            Statement stm;
            ResultSet rs1,rs2;
            double rm = 0;
            double rt = 0;
            double invalid_rows;
            String ente = null;

            Number sommeLiq_constraint;


            try {
                        //call the function to compute the total number of rows on aggiudicatari
                        rs1 = count_total_number_of_row_lotto();
                        stm = conn.createStatement();
                        int i = 1;
                        while(rs1.next())
                        {
                                    ente = rs1.getString("ente");
                                    rt = rs1.getDouble("total_rows");
                                    String query = "SELECT COUNT(*) as invalid_value FROM appalti.lotti as
                                    l LEFT JOIN appalti.lotti_aggiudicatari as l_a ON l.idCig = l_a.lotto_idCig
                                    LEFT JOIN appalti.metadati AS m ON l.metadati_urlFile = m.urlFile
                                    WHERE l_a.aggiudicatari_idAggiudicatario is null AND
                                    l.importoSommeLiquidate <> '0' AND entePubblicatore = '"+ente+"'";
                                    rs2 = stm.executeQuery(query);
                                    while(rs2.next())
                                    {
                                                rm = 0;
                                                rm = rs2.getDouble("invalid_value");
                                    }
                                    invalid_rows = consistency_function(rm, rt);
                        }
                        rs1.close();
```

```
                    stm.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
    }
//function used to count the total number of row of the 'lotto' table
    private ResultSet count_total_number_of_row_lotto(){

                    ResultSet rs = null;
                    Statement stm;

                    String query_count_total_number_of_row = "SELECT entePubblicatore as ente, count(*)
                    as total_rows FROM appalti.metadati AS m LEFT JOIN appalti.lotti AS l ON m.urlFile =
                    l.metadati_urlFile  GROUP BY entePubblicatore";

                    try {
                        stm = conn.createStatement();
                        rs = stm.executeQuery(query_count_total_number_of_row);
                    } catch (SQLException e) {
                            e.printStackTrace();
                        }
                    return rs;
            }
//function used to compute the number of row wich does not violate the constraints
    private double consistency_function(double rm, double rt)
    {
                    double div;
                    double completeness_value;

                    div = rm/rt;
                    completeness_value = 1-(div);

                    return completeness_value;
    }
```

#### 5.2.4.10 successfulTenderer_awardAmount

The metric computes the percentage of tuple that meet the constraint, that is, if a lot has a successful tenderer the award amount must be different by zero.

The algorithm is:

1. For each *entePubblicatore,* counts the number of tuples that do not meet the constraint.
2. For each *entePubblicatore*, counts the total number of tuples in the *lotti* table.
3. Compute the metric:

$$successfulTenderer\_awardAmount = 1 - \frac{rm}{rt}$$

where:

- *rm:* number of lots that have successful tenderer but the *importoAggiudicazione* attribute value is equal to zero.
- *ct:* total number of tuples of *lotti* table.

In the following code is shown the implementation of the metric.

```
//evaluation of the following integrity constraint
//when the aggiudicatario is present, the importoAggiudicazione MUST be different by zero
  public void successfulTenderer_awardAmount()
  {
            Statement stm;
            ResultSet rs1,rs2;
            double rm = 0;
            double rt = 0;
            String ente = null;
            double invalid_rows = 0;
            try {
            //call the function to compute the total number of rows on aggiudicatari
            rs1 = count_total_number_of_row_lotto();
            stm = conn.createStatement();

            while(rs1.next())
            {
                        ente = rs1.getString("ente");
                        rt = rs1.getDouble("total_rows");
                        String query = "SELECT  COUNT(*) AS invalid_value  FROM appalti.lotti as l LEFT
                        JOIN appalti.lotti_aggiudicatari as l_a ON l.idCig = l_a.lotto_idCig  LEFT JOIN
                        appalti.metadati as m ON l.metadati_urlFile = m.urlFile  WHERE
                        l_a.aggiudicatari_idAggiudicatario is not null AND l.importoAggiudicazione = '0' AND
                        entePubblicatore ='"+ente+"'";
                        rs2 = stm.executeQuery(query);
                        while(rs2.next())
                        {
                                    rm = 0;
                                    rm = rs2.getDouble("invalid_value");
                        }
                        invalid_rows = consistency_function(rm, rt);
            }

      rs1.close();
      stm.close();
      } catch (SQLException e) {
                  e.printStackTrace();
          }
}
//function used to count the total number of row of the lotti table
  private ResultSet count_total_number_of_row_lotto(){

                  ResultSet rs = null;
                  Statement stm;

                  String query_count_total_number_of_row = "SELECT entePubblicatore as ente, count(*)
                  as total_rows FROM appalti.metadati AS m LEFT JOIN appalti.lotti AS l ON m.urlFile =
                  l.metadati_urlFile  GROUP BY entePubblicatore";

                  try {
                        stm = conn.createStatement();
                        rs = stm.executeQuery(query_count_total_number_of_row);
                      } catch (SQLException e) {
                                  e.printStackTrace();
                              }
                      return rs;
            }
```

```
//function used to compute the number of row wich does not violate the constraints
  private double consistency_function(double rm, double rt)
  {
                    double div;
                    double completeness_value;

                     div = rm/rt;
                    completeness_value = 1-(div);

                   return completeness_value;
  }
```

*Chapter 6*

# 6. Analysis of the Results

In this chapter we analyse the results of the metrics calculated on data provided by the Italian Universities. The results, of each computed dimension, are presented as graphs showing the quality of the data published by each contracting authority. For each of them we will discuss the results and the causes in terms of values extracted by the source file. That is, we will describe the dimensions in terms of values found in the XML file that then are transformed to compute the metric as explained in chapter 3. To conclude, will be shown the Universities classification that provides us the information about the best universities in terms of quality of published data.

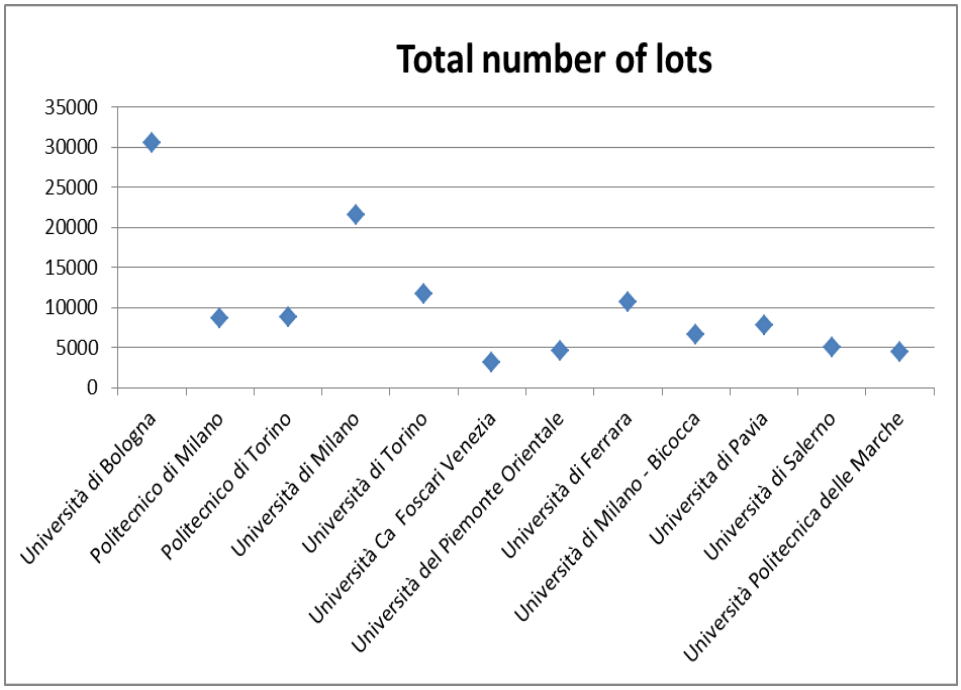



**Figure 22**:Total number of lots published by each University

To better understand and evaluate the results of the evaluation, a key information to provide is the number of lots contained in the summary tables published by each University in the year 2014 and on which the analysis in performed. Figure 20 shows the number of lots published by the different Universities. The University of Bologna published for the 2014 more than 30000 lots distributed in many summary tables. It is followed by the University of Milano with a number of published lots around 20000. The University of Venezia is the one that published less number of lots. The big

difference on the number of published lots by each university depends not necessarily on a failure by the universities themselves but, instead, it depends on their size. It can be assumed that a university of small dimensions and with a smaller number of students, has a number of public contracts lower than a university of larger dimensions and with a greater number of students. Knowing the number of lots on which the analysis was carried out, we can continue with the analysis of the obtained results. For each attribute of each table will be shown the result of the calculated dimensions.

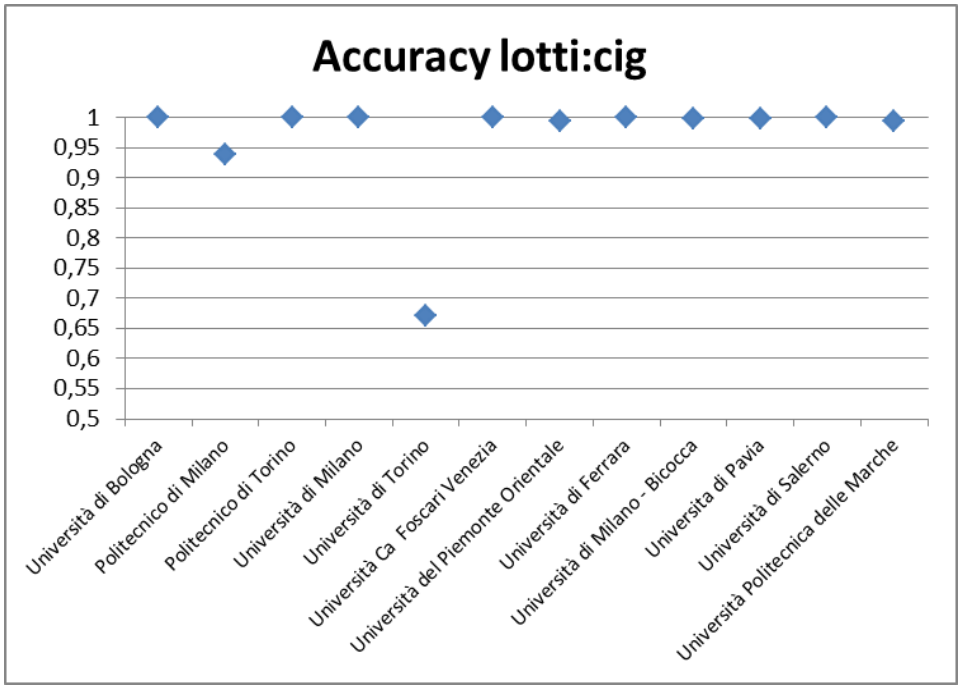



**Figure 23:** Accuracy of cig in lotti table

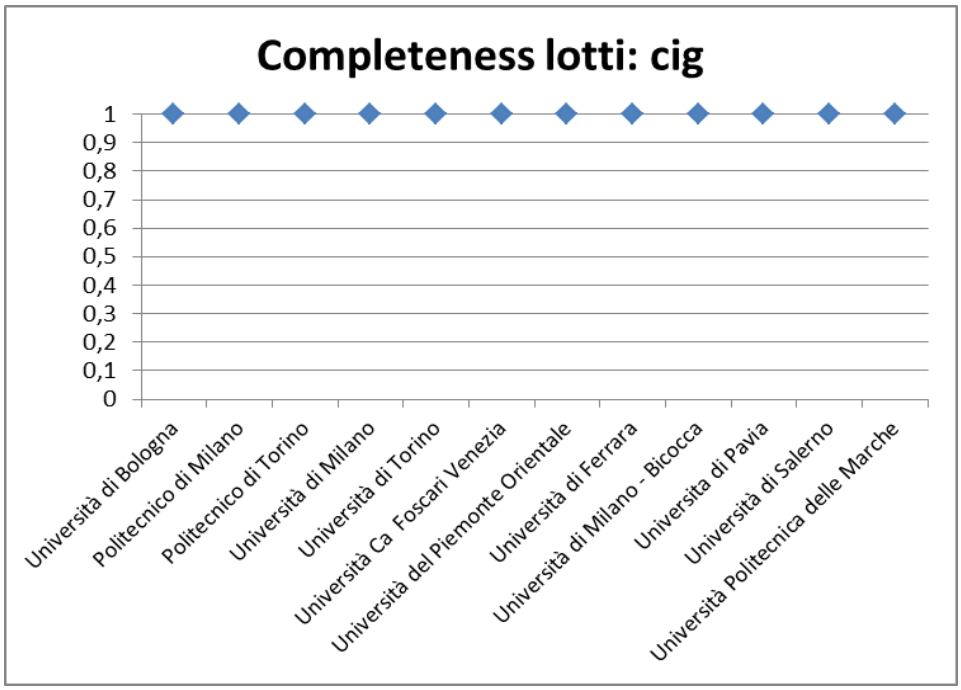



**Figure 24:** Completeness of cig in lotti table

The cig is a crucial information because it provides the unique id of each contract. Given its importance, the accuracy and completeness on cig should be very high. From figures 23 and 24, we can see that the percentage of complete cells is rather high for all the universities but, for some of them the percentage of accurate cells is subject to variation. This means that, although the element cig is present in all the lots, in some certain cases it is outside the domain, that is, has a number of digits either other than 10 or is blank. The University of Torino publishes summary tables that have 100% *cig* completeness, that is, the 100% of lots have the cig element but about 30% of them are out of domain that is not useful to uniquely identify a contract.

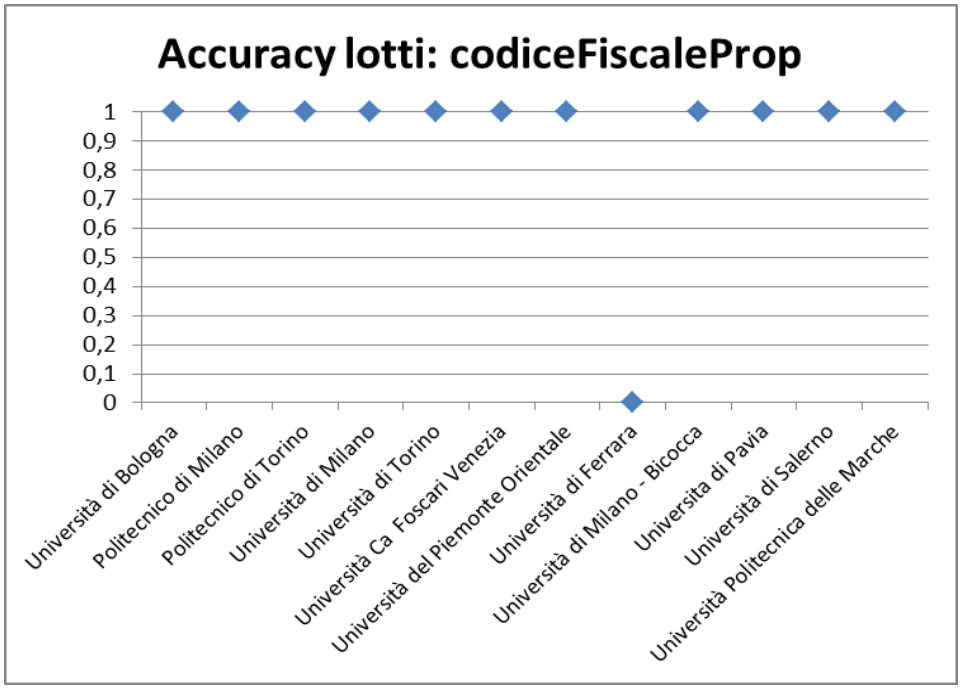



**Figure 25:** Accuracy of codiceFiscaleProp in lotti table

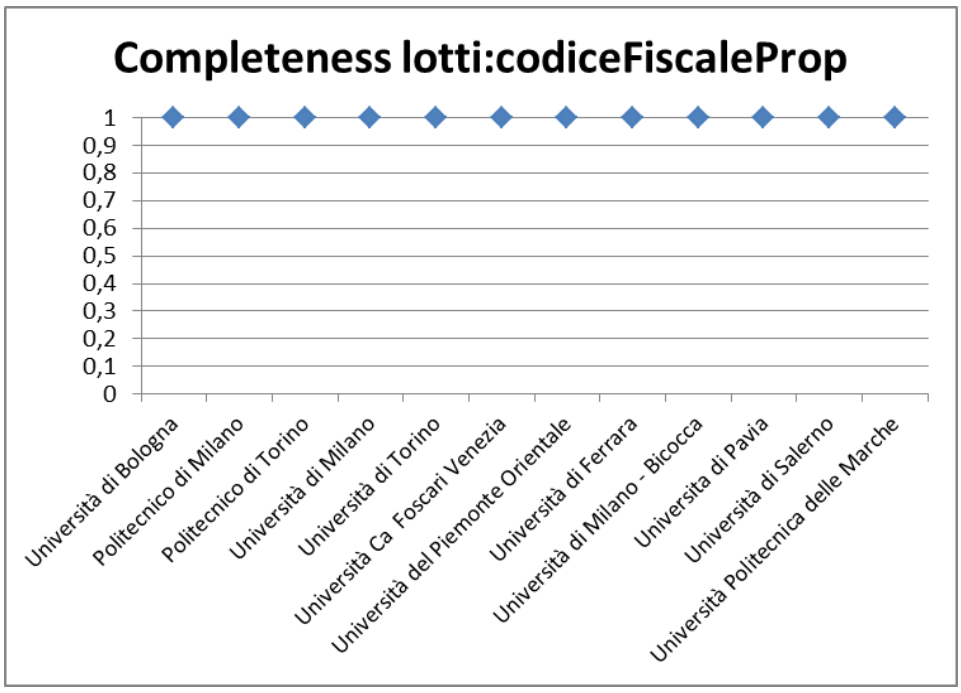



**Figure 26:** Completeness of codiceFiscaleProp in lotti table

Figures 25 and 26 shows that the percentage of complete cells and the percentage of accurate cells is the 100% for all the Universities except the University of Ferrara which have the 100% of complete cells and the 0% of accurate cells. This is due to the fact the 100% of codiceFiscaleProp elements is always present in the lots, extracted by the original file published by the University of Ferrara, but it is always empty.

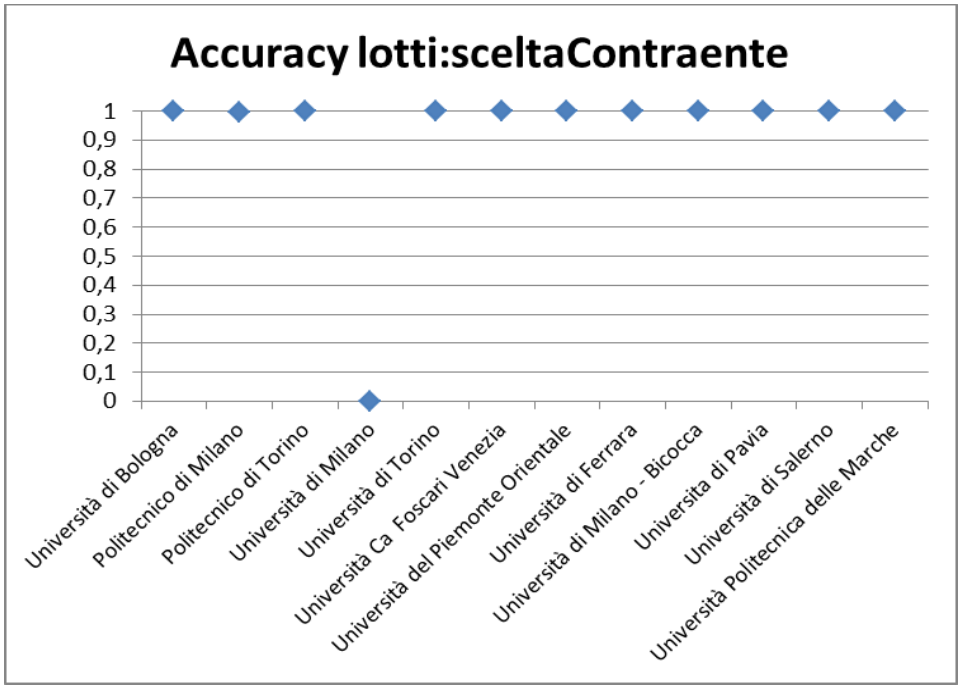



**Figure 27:** Accuracy of sceltaContraente in lotti table

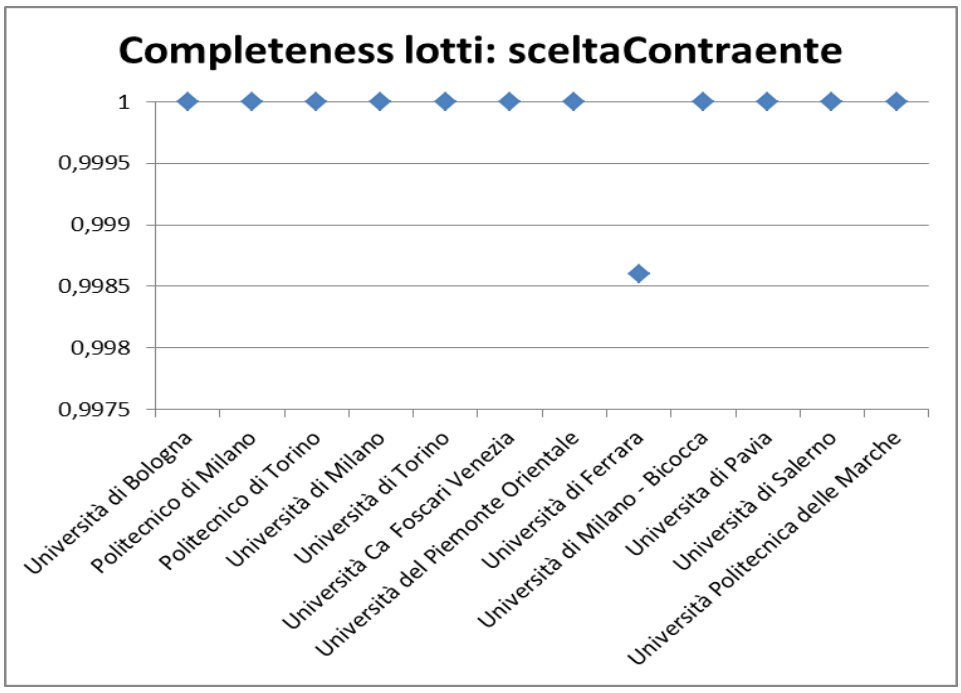



**Figure 28:** Completeness of sceltaContraente in lotti table

The *sceltaContraente* is a crucial information because it indicates the procedure of contractor selection and it is often used by the authorities in identifying illegal award of a contract. A good level of accuracy and completeness of this information is fundamental to increase transparency of public contracts. From figure 27 and 28, we notice that universities pay much attention to providing this information, in fact, the percentage of complete and accurate cells for the *sceltaContraente* attribute is the 100% for almost all universities. The only one that is different from the other is the University of Milano where the completeness is 100% but the accuracy is 0% because *sceltaContraente* element in the original file is always present but it is empty.

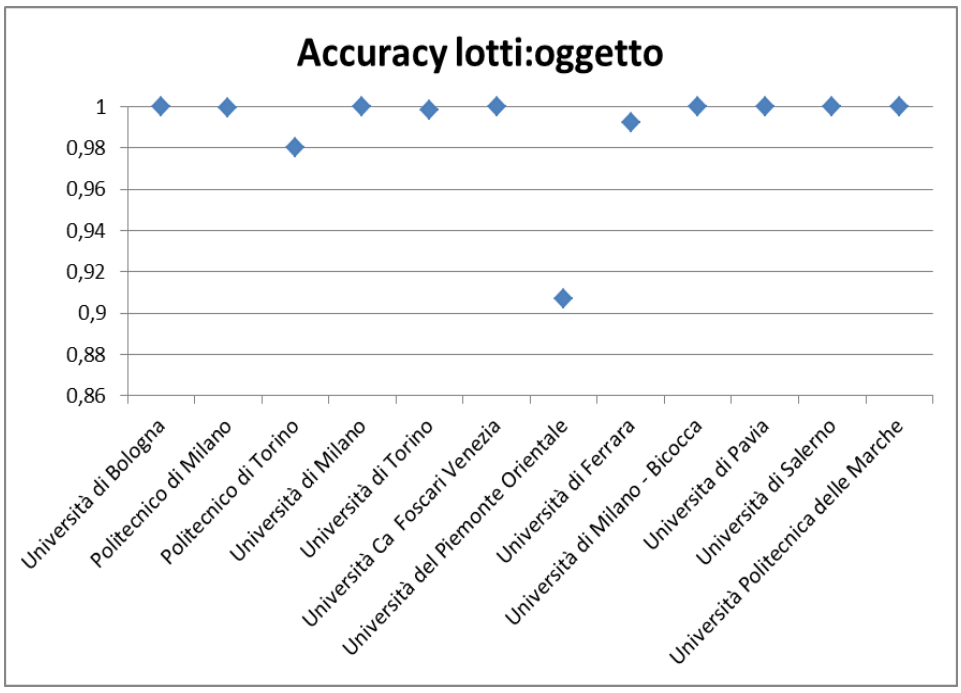



**Figure 29:** Accuracy of oggetto in lotti table

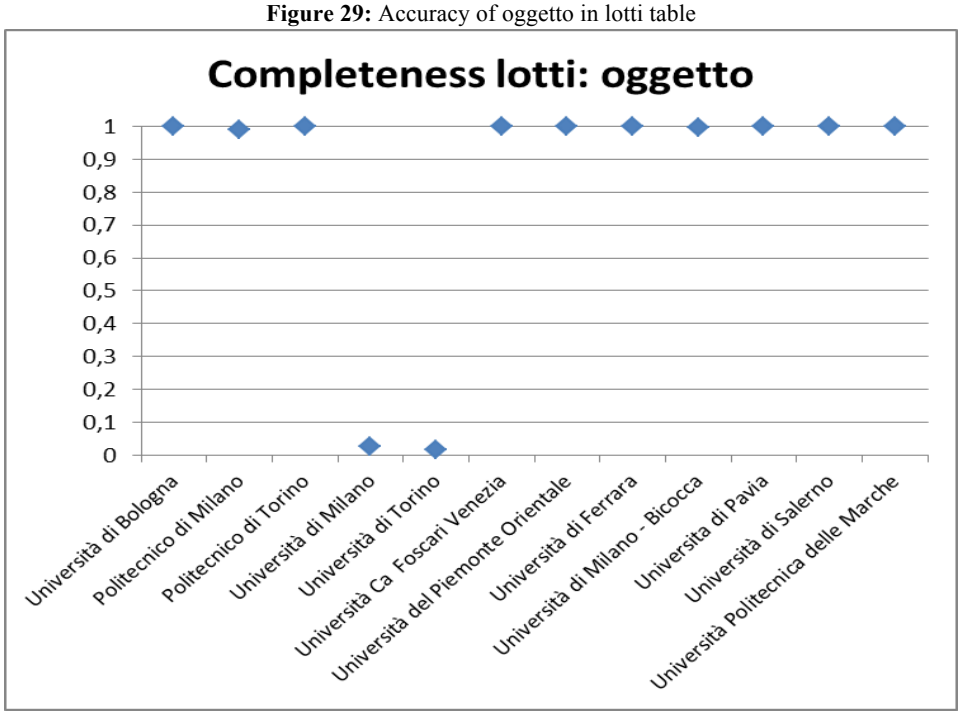



**Figure 30:** Completeness of oggetto in lotti table

The *oggetto* provides information about the subject of the contract. Figures 29 and 30 show that the percentage of accurate cells suffer of a slight variation for the University of Piemonte Orientale and for Politecnico di Torino. The percentage of complete cells is rather low for the University of Torino and University of Milano which have both around 98% of incomplete cells that means that elements in the original file are either missing or present but empty.

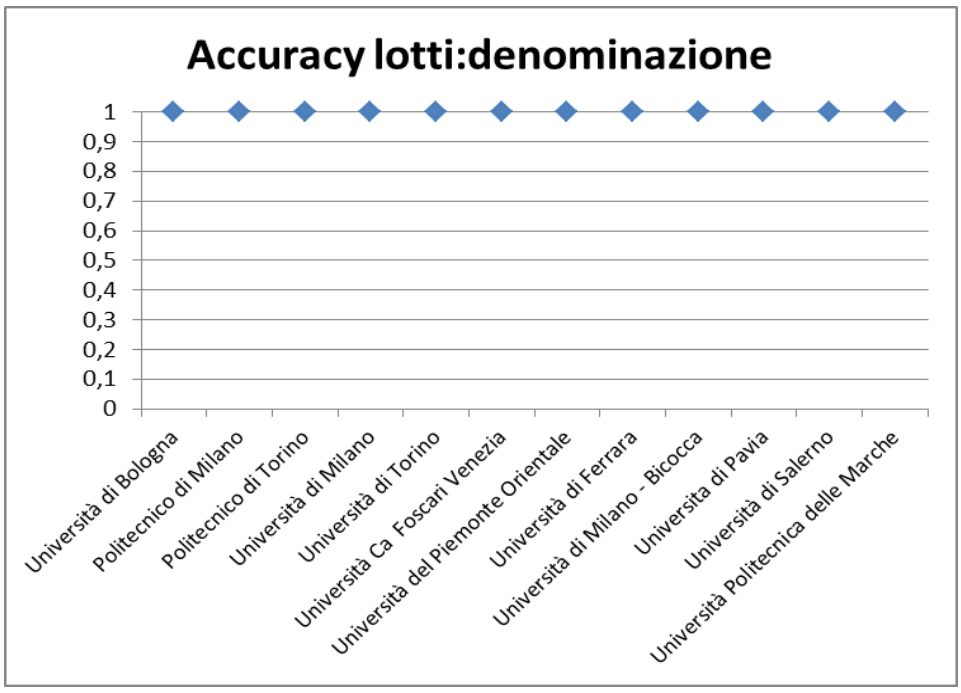



**Figure 31:** Accuracy of denominazione in lotti table

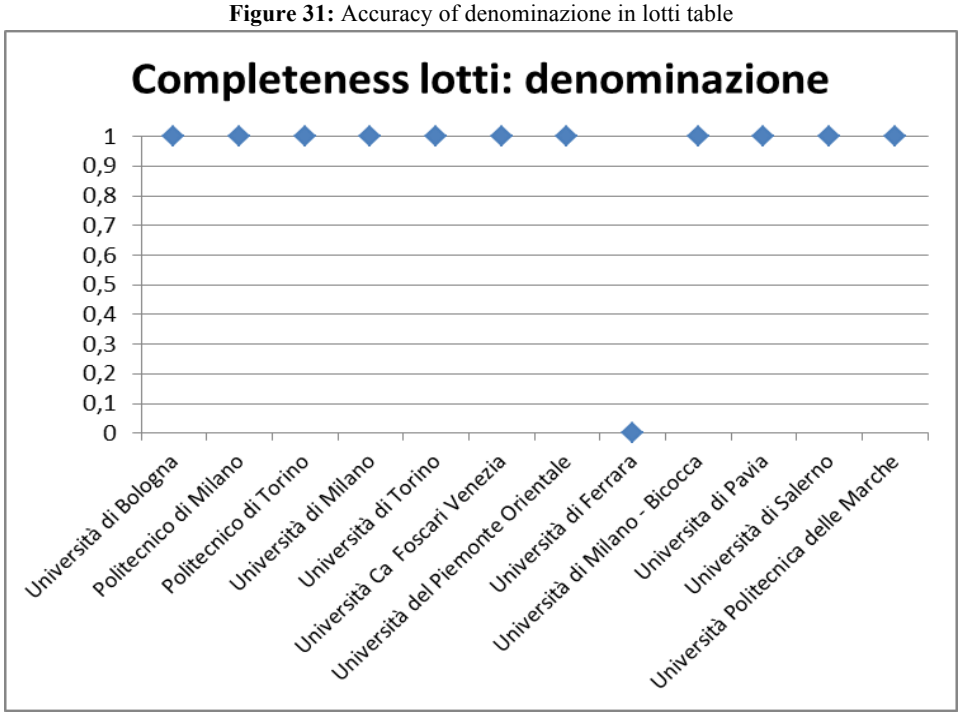



**Figure 32:** Completeness of denominazione in lotti table

The percentage of accurate and complete cells is 100% for all the universities except for the University of Ferrara which has 0% completeness of *denominazione* attribute. This means that all the universities have the 100% of *denominazione* elements in the original files with a length less than 250. The University of Ferrara has 100% of incomplete cells this is because in all lots of the source file, the *denominazione* element is present but it is empty.

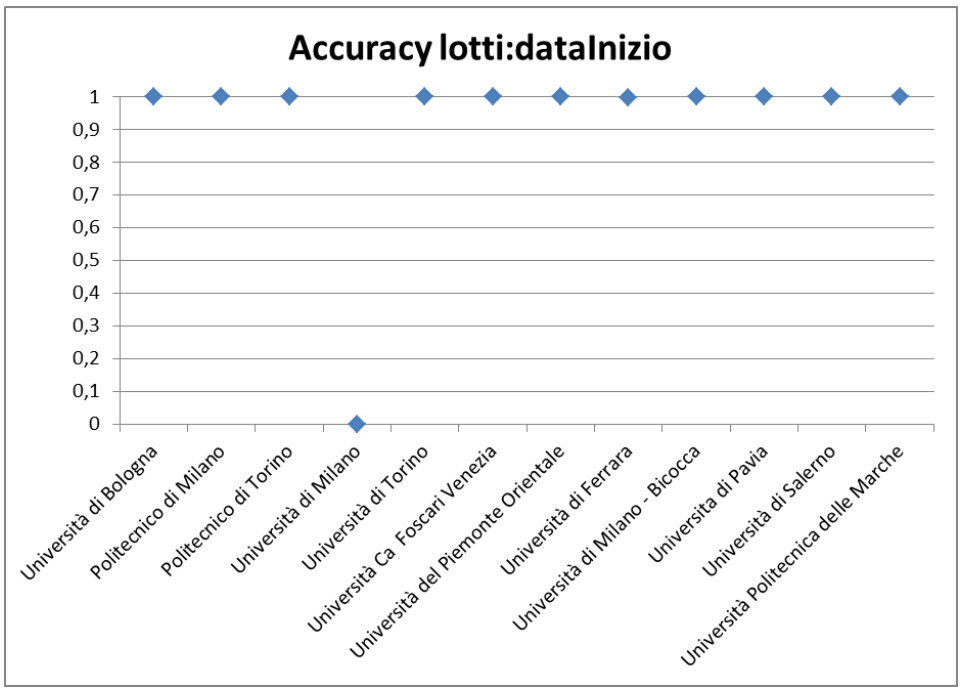



**Figure 33:** Accuracy of dataInizio in lotti table

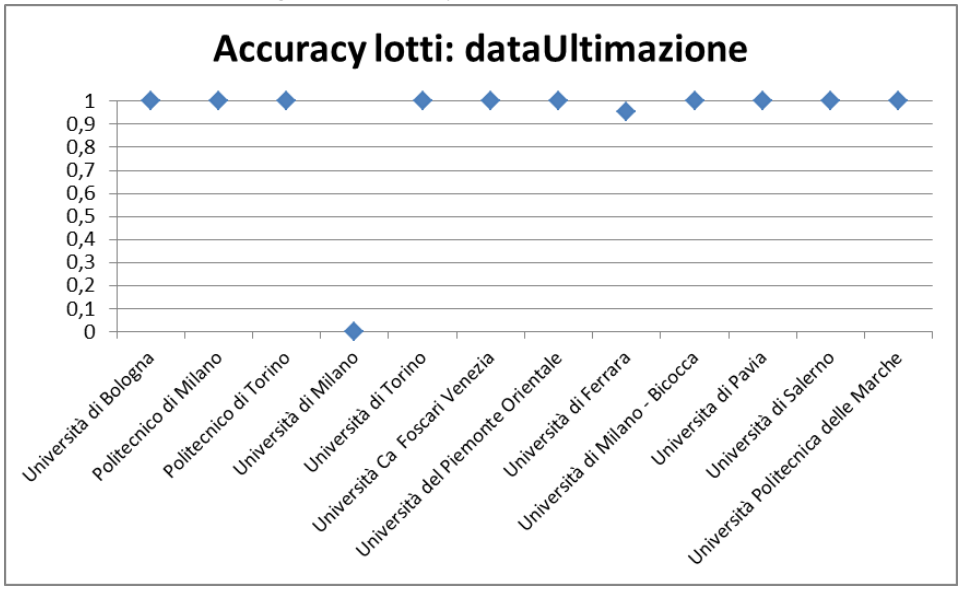



**Figure 34:** Accuracy of dataUltimazione in lotti table

The completeness dimension in not computed for the *dataInizio* and *dataUltimazione* since the XML Schema fixes a minimum occurrence for them equal to zero. Thus, *dataInizio* and *dataUltimazione* elements could be absent in the source file but this would not be a completeness error.

Concerning on *dataInizio*, the percentage of accurate cells is about 100% for all the universities except for the University of Milano in which the *dataInizio* is always present within each lot but it is blank. The same is for the evaluation of the *dataUltimazione* for the University of Milano where the *dataUltimazione* elements, in the XML files, is present is each lot but it is always empty.

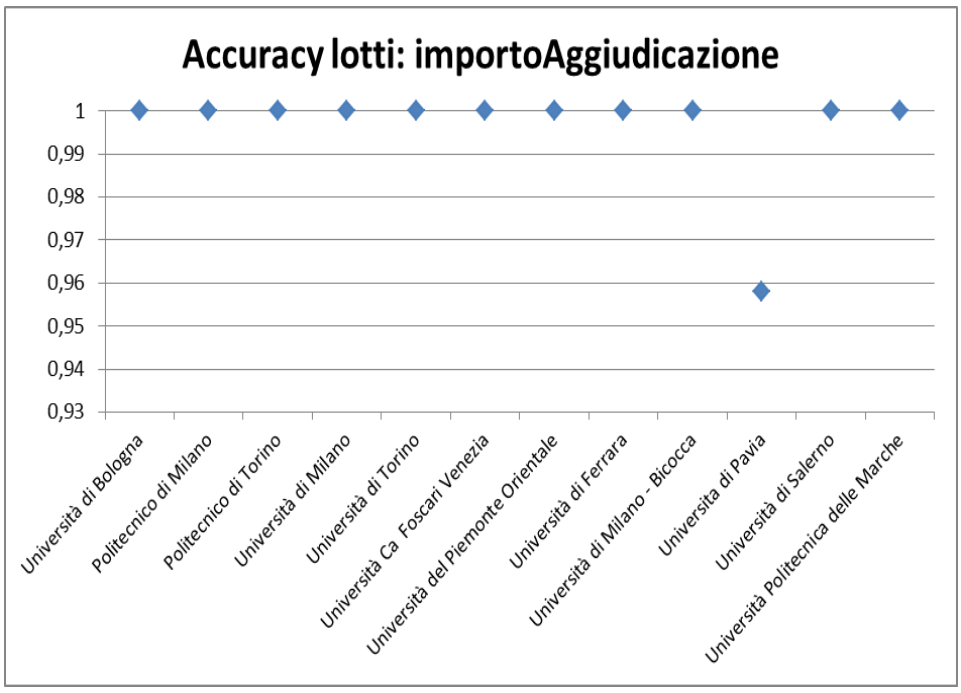



**Figure 35:** Accuracy of importoAggiudicazione in lotti table

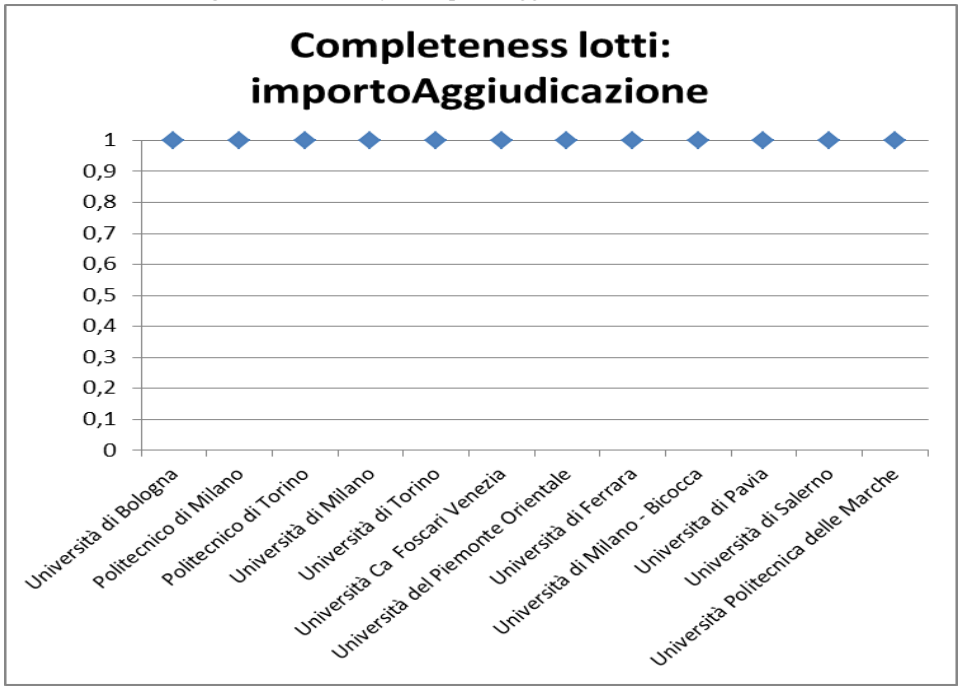



**Figure 36:** Completeness of importoAggiudicazione in lotti table

Figures 35 and 36 show that, in general, the accuracy and the completeness of *importoAggiudicazione* attribute is rather high, the only variation is given by the University of Pavia which has about 5% of cells that are not accurate, this means that the value, retrieved by the XML file, of 5% of *importoAggiudicazione* elements is either an out of domain decimal or empty.

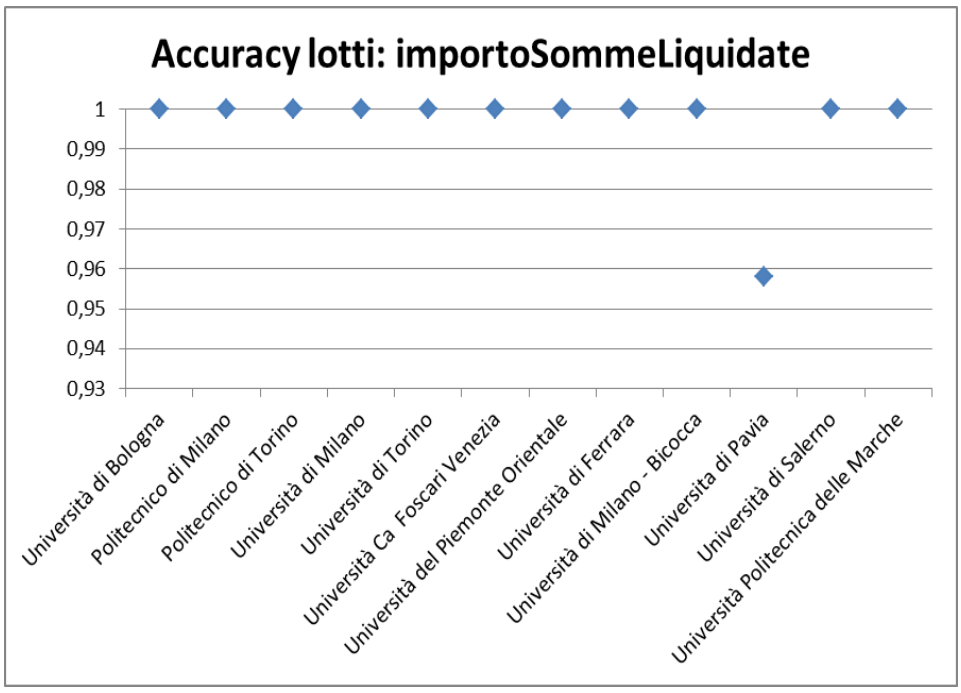



**Figure 37:** Accuracy of importoSommeLiquidate in lotti table

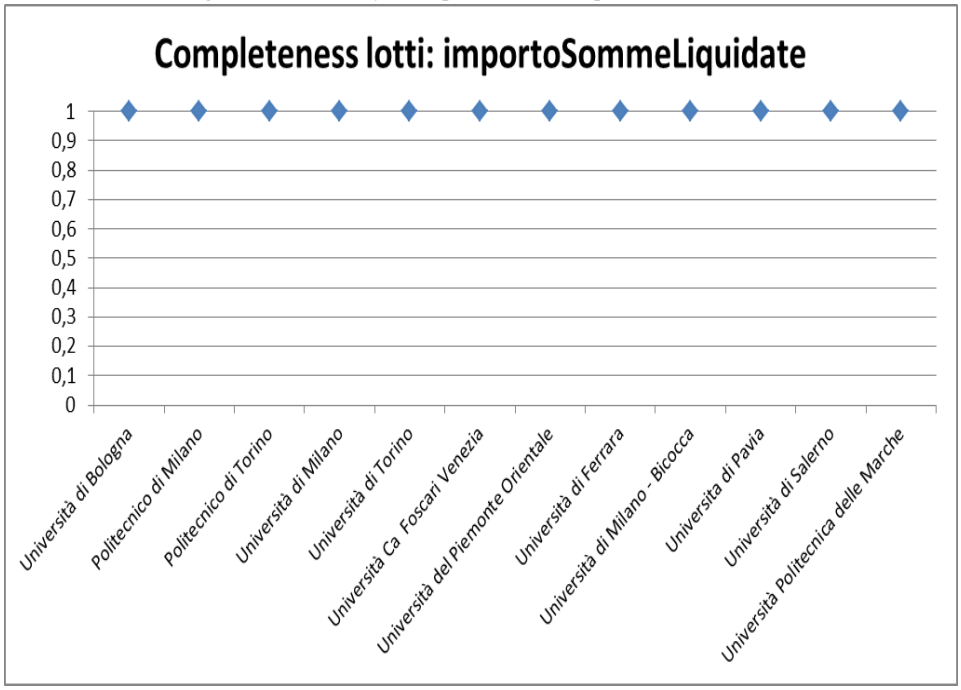



**Figure 38:** Accuracy of importoSommeLiquidate in lotti table

As for the accuracy and the completeness of *importoAggiudicazione* attribute, even for the *importoSommeLiquidate* attribute a slight variation is given by the University of Pavia which has about 5% of inaccurate cells. The reason is that the 5% of *importoSommeLiquidate* elements in the source files, provided by the University of Pavia, are out of domain or empty.

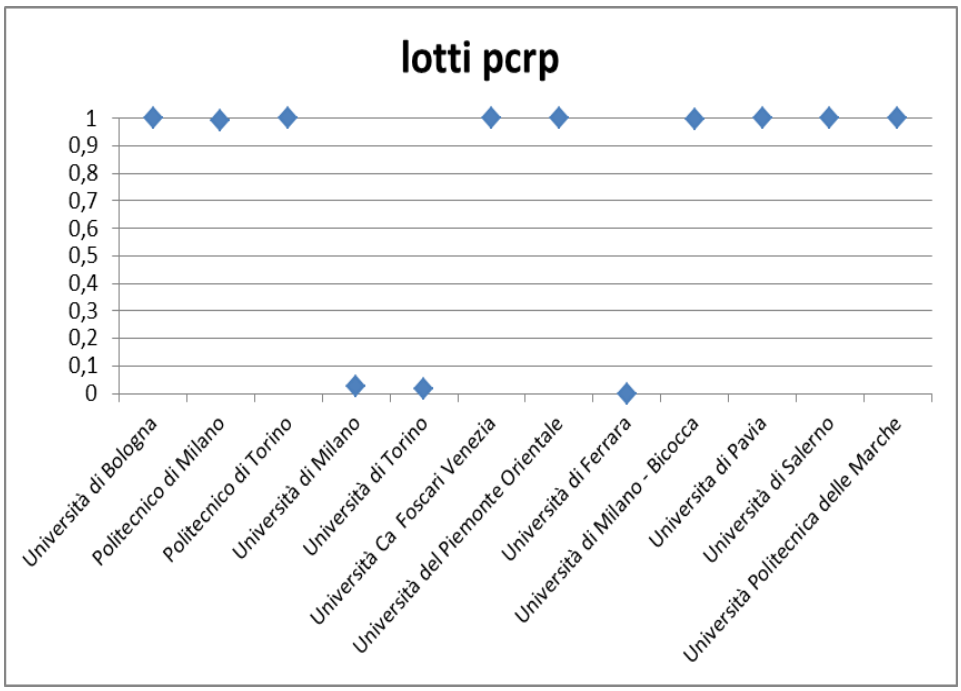



**Figure 39:** Completeness on lotti tuples

The Figure above shows the completeness on tuples. It gives information of how incomplete values are distribute in *lotti* table. The worst University in terms of completeness with regard to the attributes in *lotti* table is the University of Ferrara followed by Universities of Torino and Milano. For all the other universities the completeness on the attribute of lotti table is close to 100%.

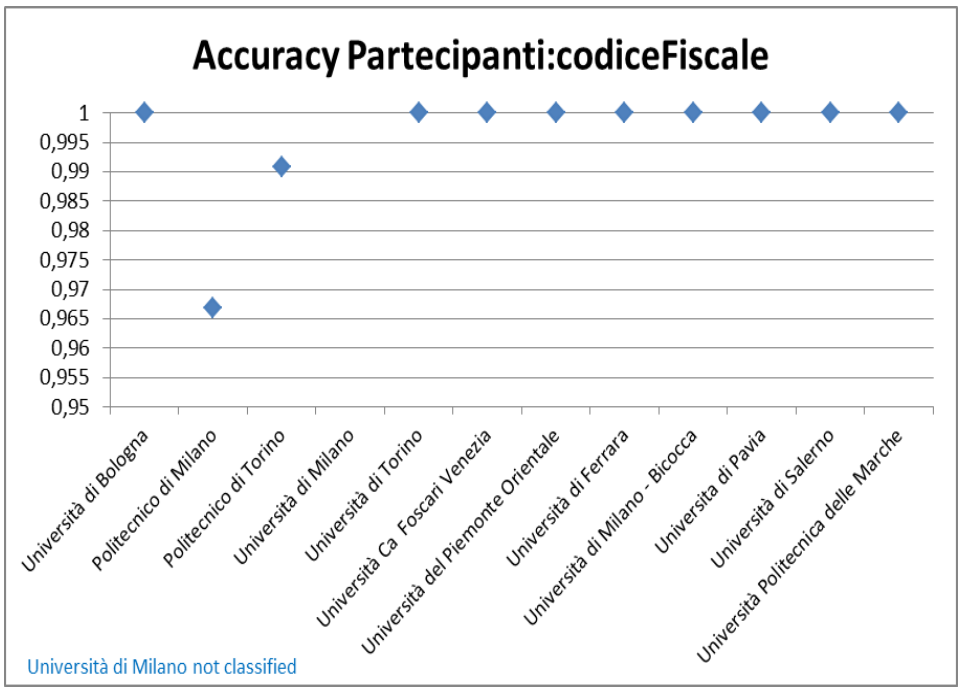



**Figure 40:** Accuracy of codiceFiscale in partecipanti table

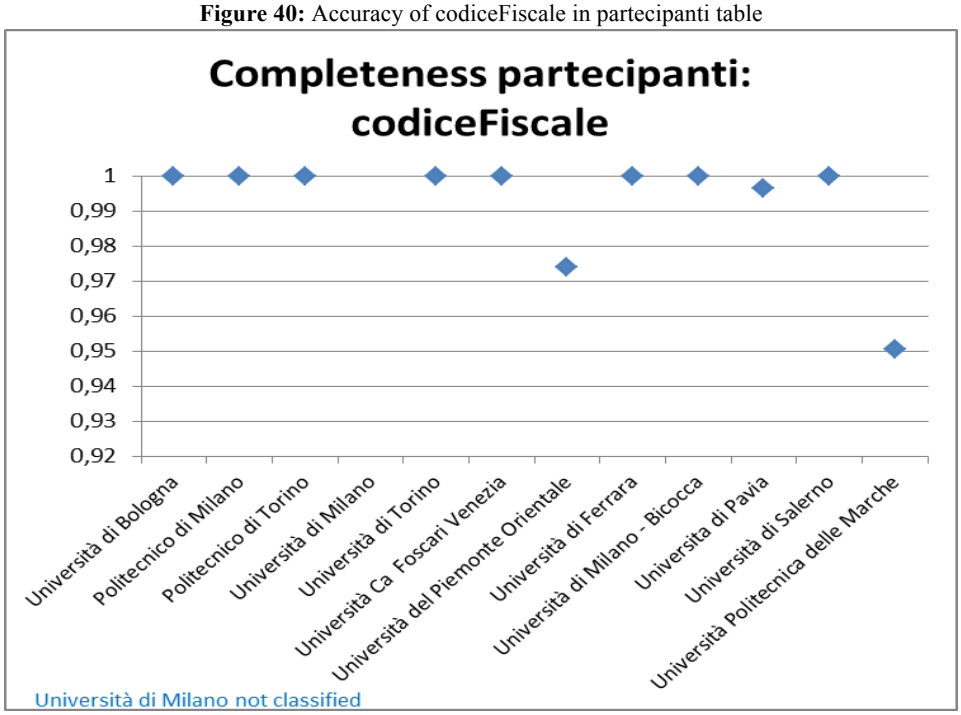



**Figure 41:** Completeness of codiceFiscale in partecipanti table

Figure 40 and 41 shows and important aspect. The accuracy and completeness for the *codiceFiscale* (and for the all the attributes of the *partecipanti* table) is not computed for the University of Milano. In the summary tables of all the files analysed for the University of Milano there aren't participant, that is, all the lots analysed have a successful tenderer but neither of them have information about the participants. In general, both the accuracy and completeness of the *codiceFiscale* of *partecipanti* table is high. Politecnico di Milano has about the 4% of inaccurate cells, that is, the

values extracted by the XML source files are either other then 11 or 16 digits or empty.

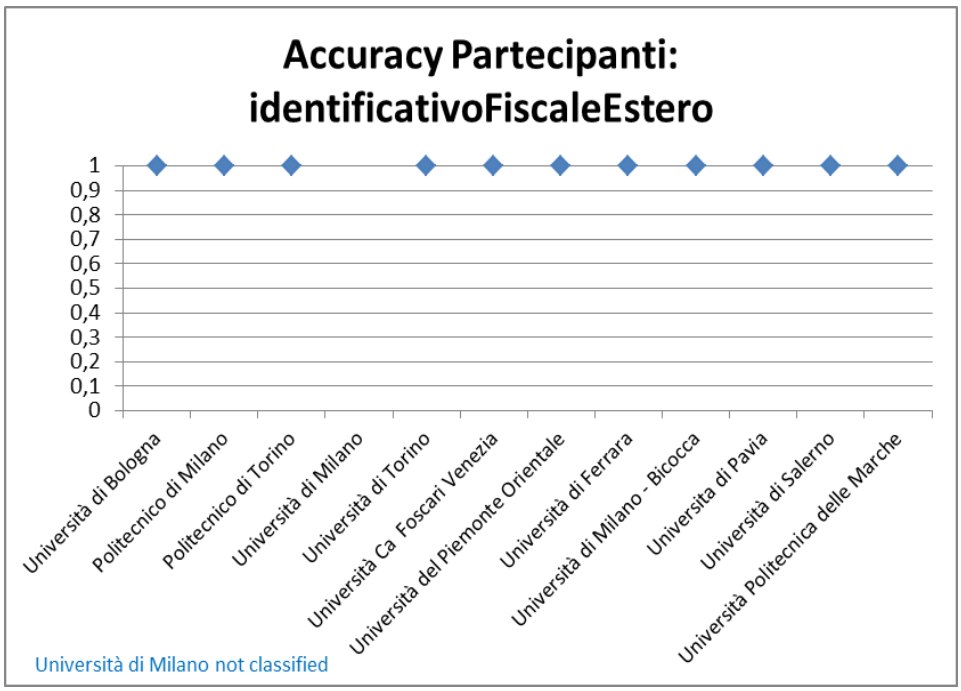



**Figure 42:** Accuracy of identificativoFiscaleEstero in partecipanti table

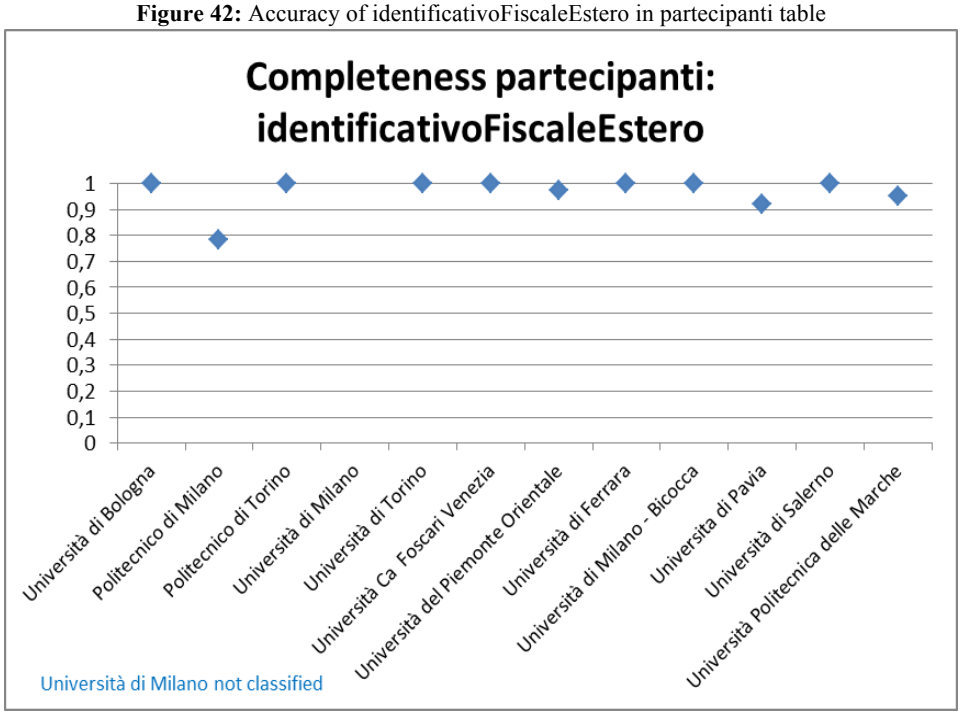



**Figure 43:** Completeness of identificativoFiscaleEstero in partecipanti table

Figure 43 shows that identificativoFiscaleEstero completeness is quite variable among universities. The worst percentage of complete cells is given by the Politecnico di Milano that has about the 22% of incomplete cells. This percentage takes into account the *identificativoFiscaleEstero* elements of each participant of the analysed files that are not present or are present but empty.

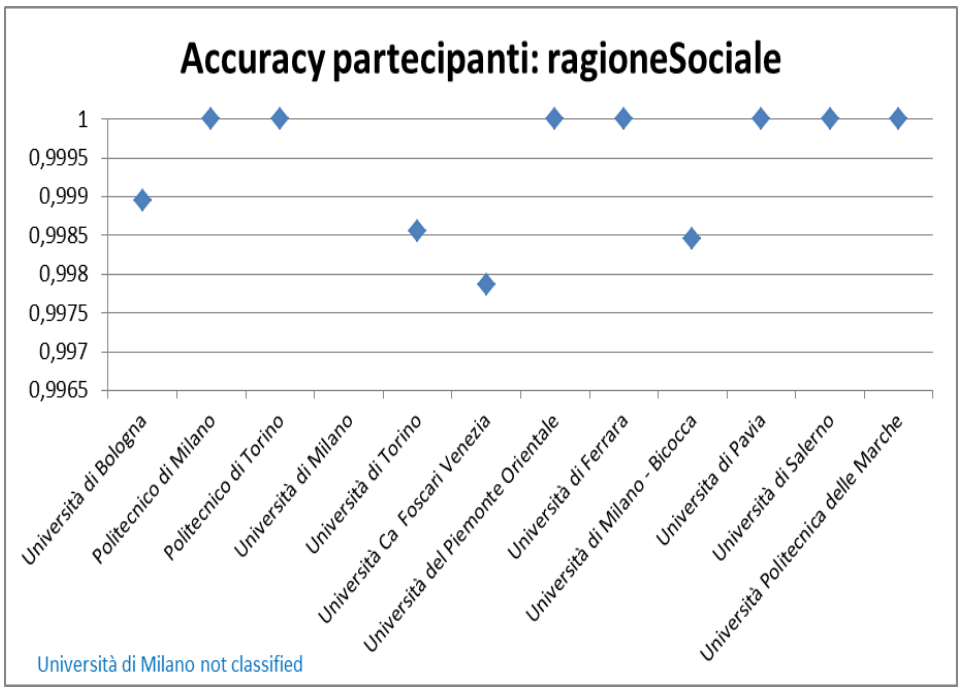



**Figure 44:** Accuracy of ragioneSociale in partecipanti table

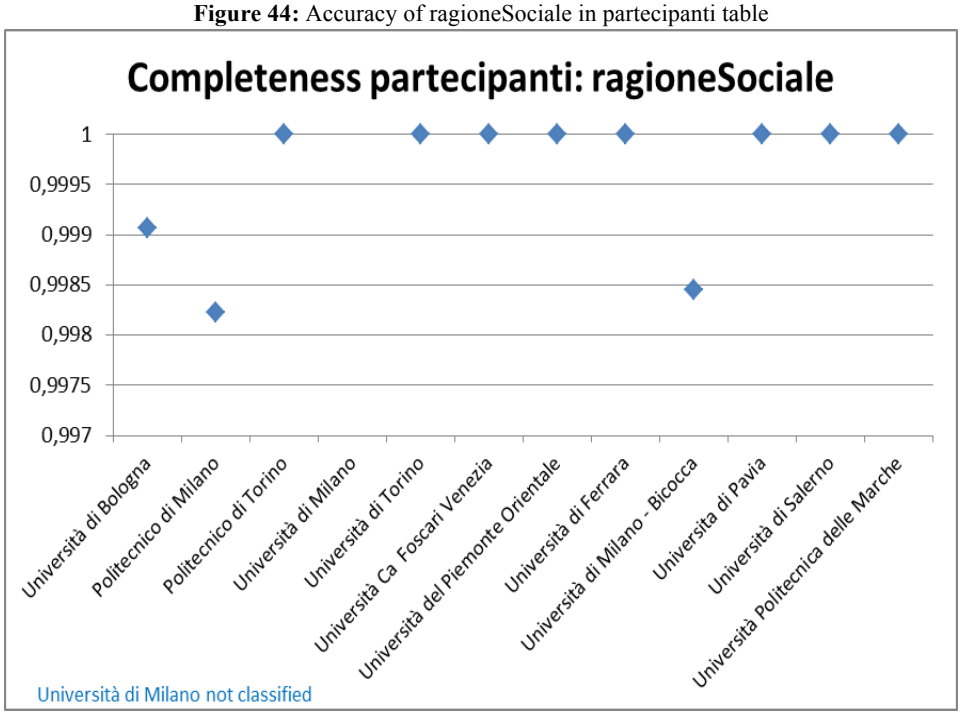



**Figure 45:** Completeness of ragioneSociale in partecipanti table

The percentage of accurate and complete cells for *ragioneSociale* attribute is around 100% for all universities except for the University of Milano which is not classified.

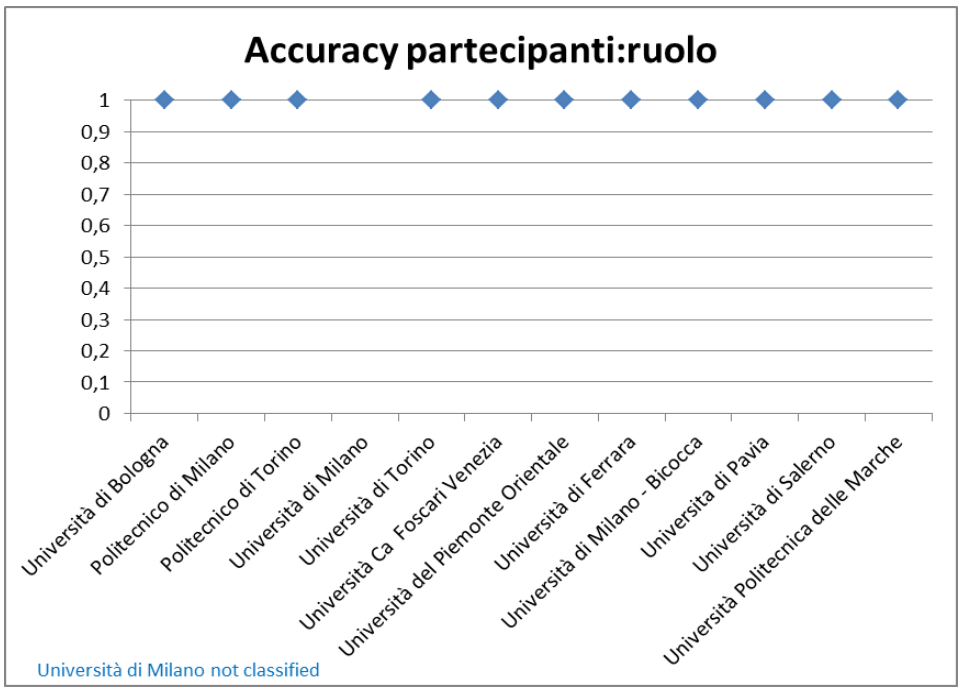



**Figure 46:** Accuracy of ruolo in partecipanti table

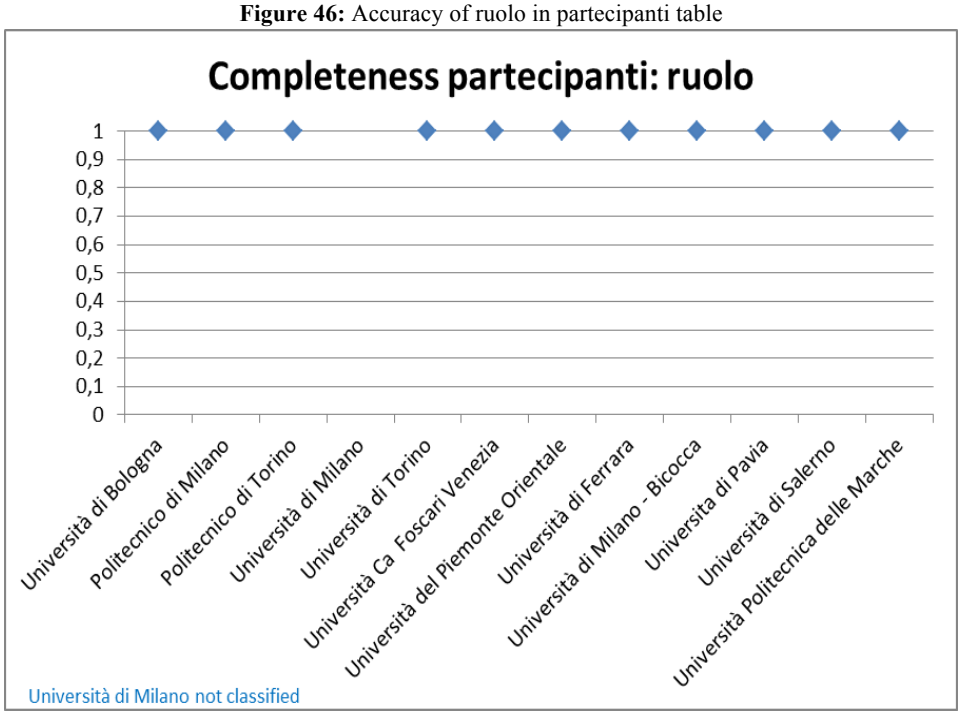



**Figure 47:**Completeness of ruolo in partecipanti table

The percentage of correct and complete cells with regard to *ruolo* attribute is the 100% for all Universities except the University of Milano that is not evaluated.

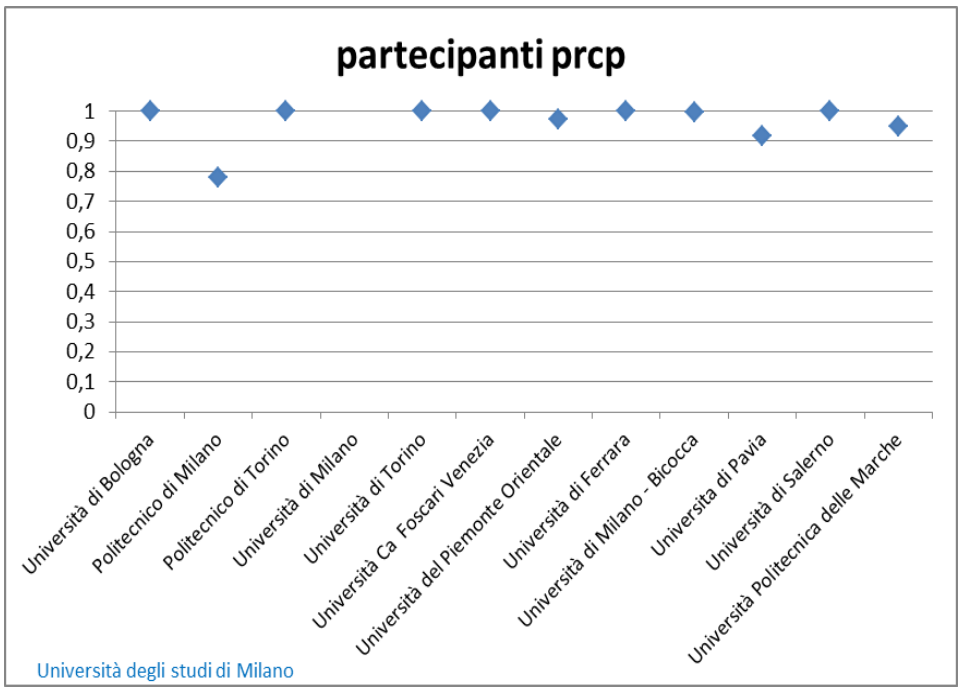



**Figure 48:**Completeness of partecipanti tuples

Figure 48 shows the completeness of tuples of *partecipanti* table. The percentage of tuples with incomplete attribute is variable for all the universities. The worst in term of completeness on the *partecipanti* table is the Politecnico di Milano while the University of Milano is still not classified.

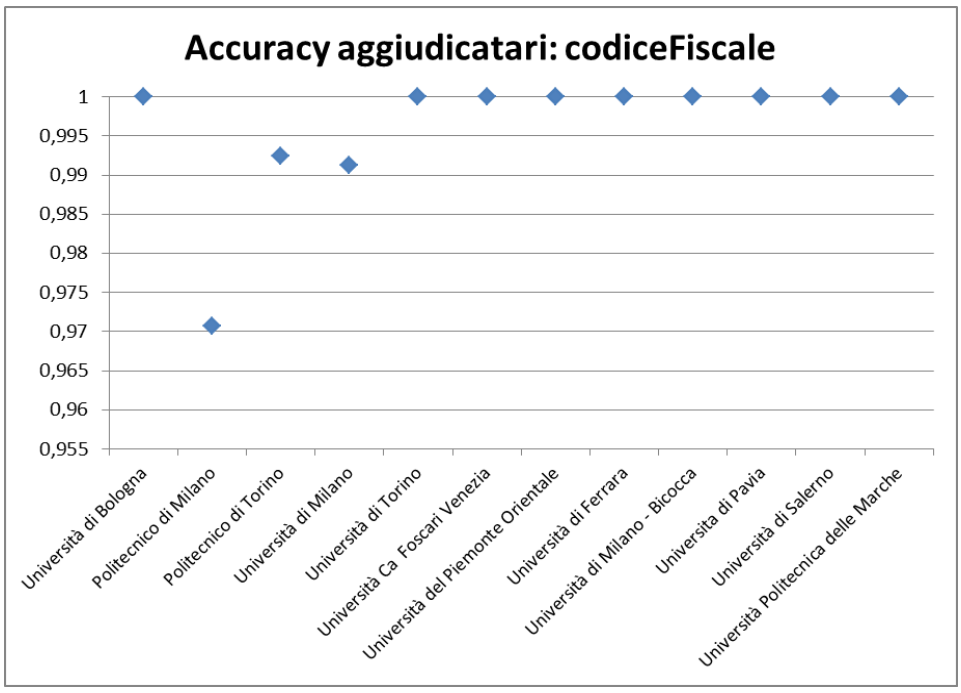



**Figure 49:**Accuracy of codiceFiscale in aggiudicatari table

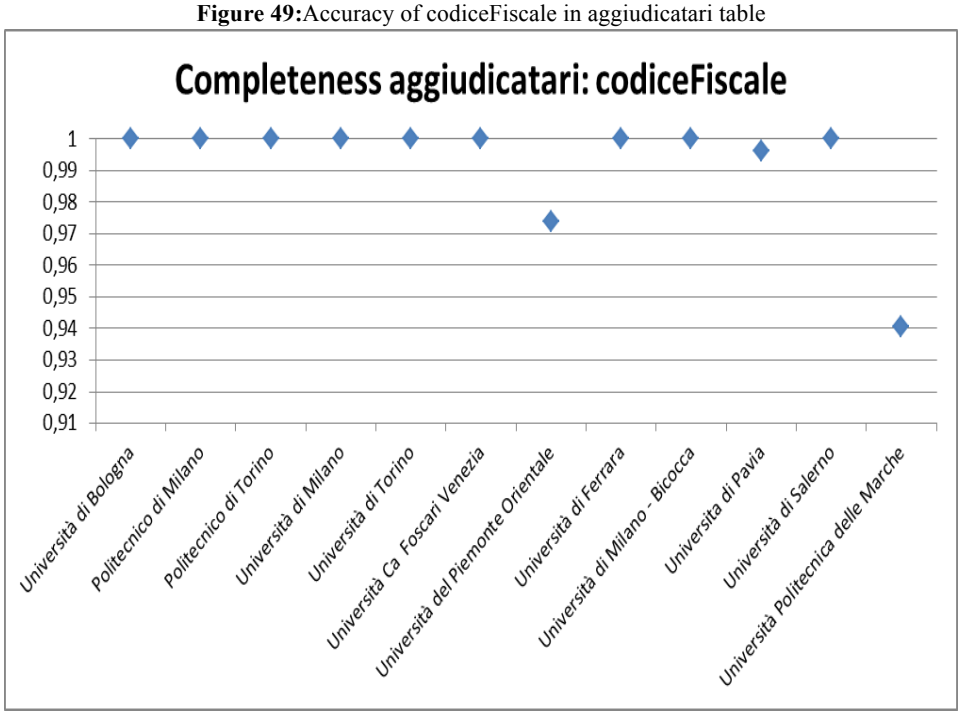



**Figure 50:**Completeness of codiceFiscale in aggiudicatari table

The percentage of accurate and complete cells for the *codiceFiscale* attribute of the *aggiudicatari* table is around 100% for all the universities. The Politecnico di Milano has a lower accuracy with the respect to the other, due to values of *codiceFiscale* elements that are either other than 11 and 16 digits or empty. The University Politecnica delle Marche has a lower completeness than the other universities because the *codiceFiscale* elements of some successful tenderer in the lots are not present. In

general the completeness and accuracy of the *codiceFiscale* of *aggiudicatari* table is high.

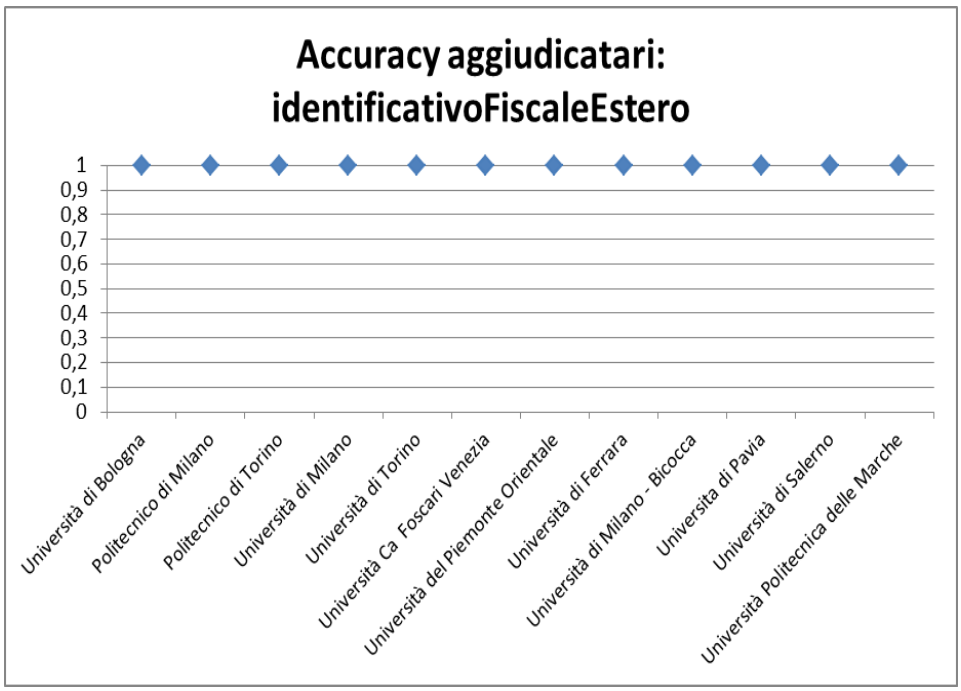



**Figure 51:**Accuracy of identificativoFiscaleEstero in aggiudicatari table

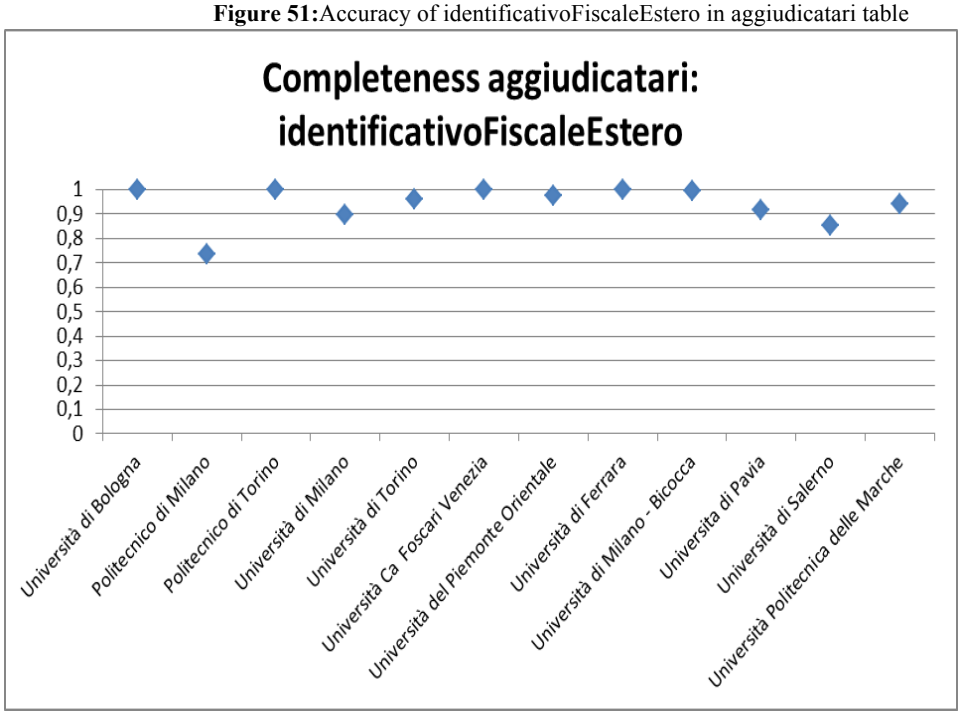



**Figure 52:**Completeness of identificativoFiscaleEstero in aggiudicatari table

Even for the completeness of the *identificativoFiscaleEstero* of *aggiudicatari* table, as for *partecipanti* table, the percentage of each university is variable. The worst university in terms of the completeness of *identificativoFiscaleEstero* attribute of *aggiudicatari* table is the Politecnico di Milano which has about the 27% of incomplete cells which correspond to successful tenderers in the file that either do not have the *identificativoFiscaleEstero* or it is empty.

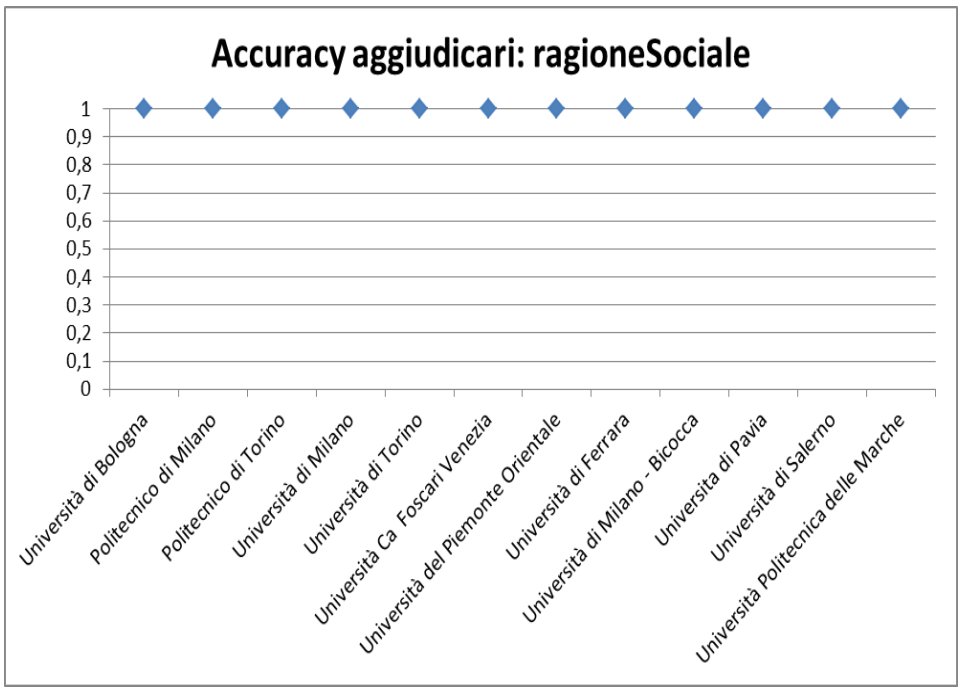



**Figure 53:**Accuracy of ragioneSociale in aggiudicatari table

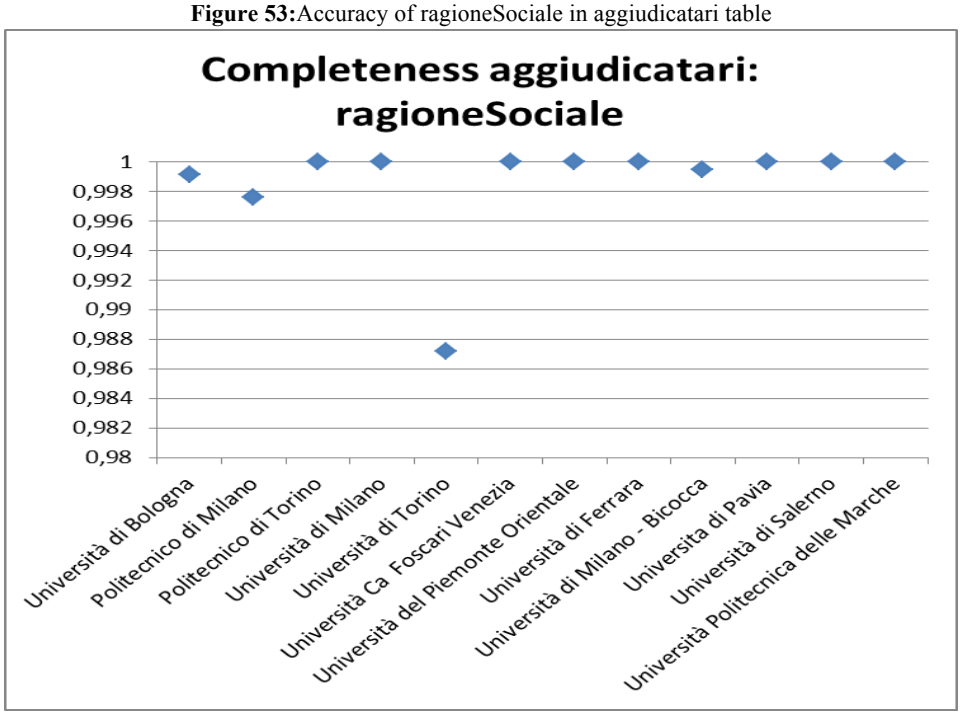



**Figure 54:**Completeness of ragioneSociale in aggiudicatari table

The percentage of accurate and complete cells of *ragioneSociale* attribute of *aggiudicatari* table is about the 100% for all the universities.

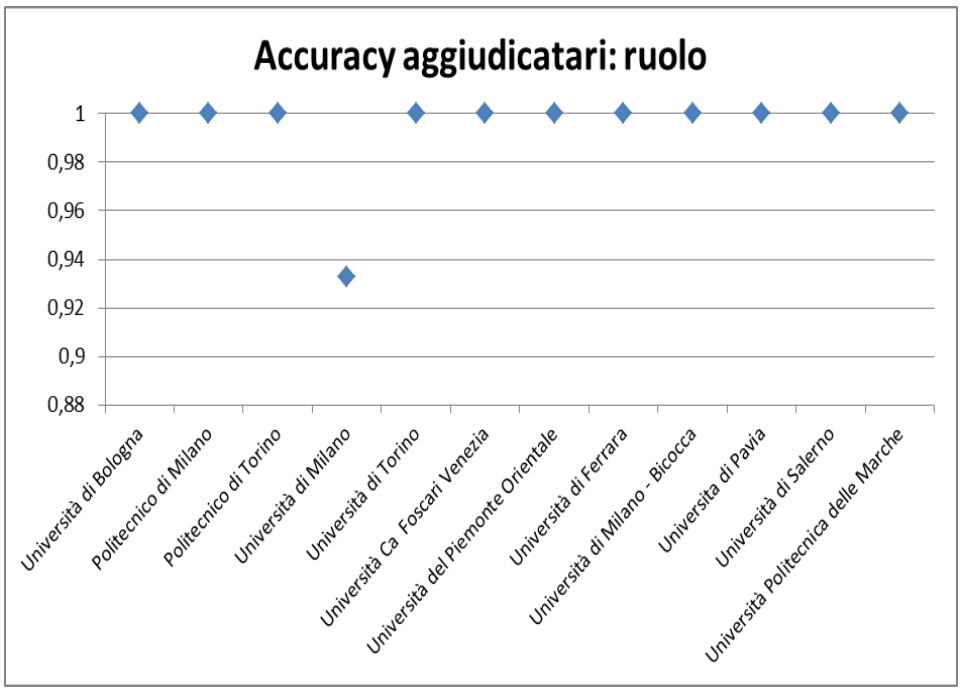



**Figure 55:**Accuracy of ruolo in aggiudicatari table

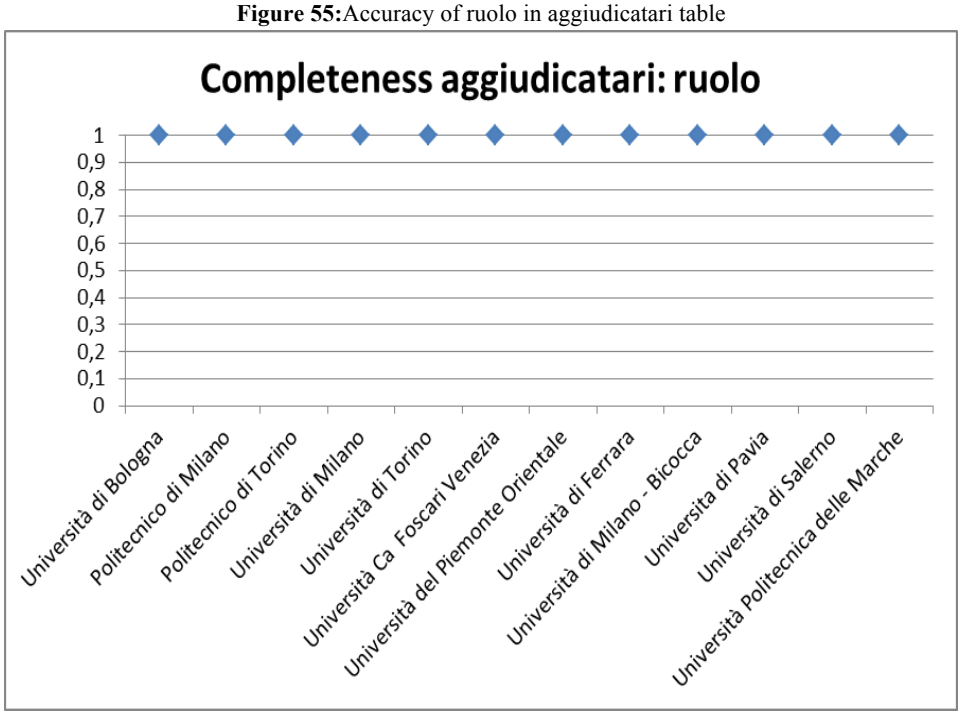



**Figure 56:**Completeness of ruolo in aggiudicatari table

The percentage of accurate and complete cells of *ruolo* attribute of *aggiudicatari* table is exactly the 100% for all the universities except for the University of Milano which has about the 7% of not accurate cells which correspond to successful tenderers which are part of a group but have *ruolo* elements that are either not in domain or are empty.

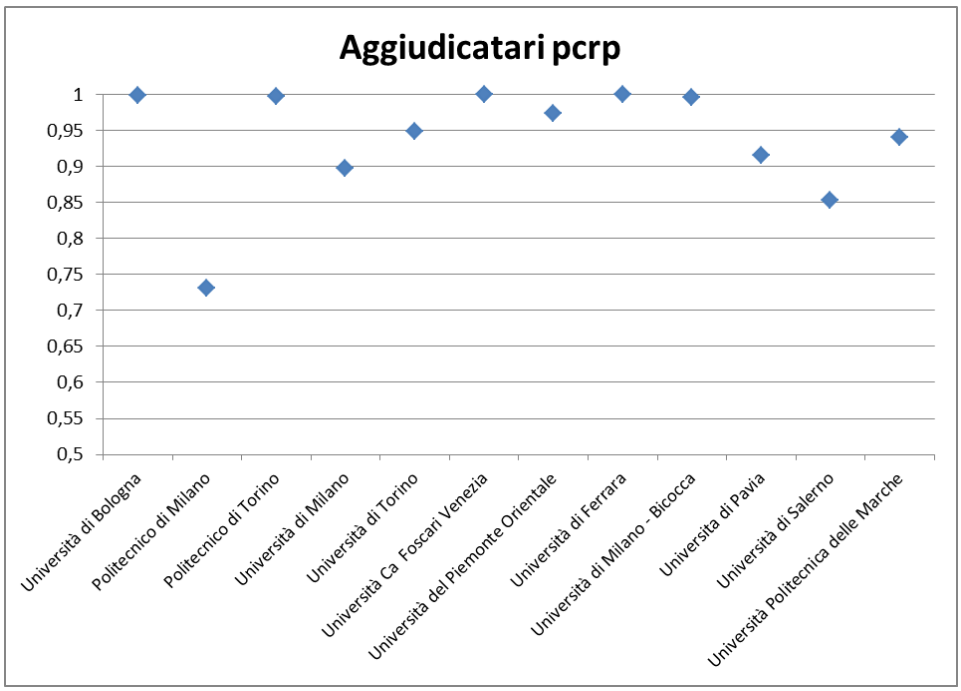



**Figure 57:**Completeness of aggiudicatari tuples

The percentage of complete tuples in the *aggiudicatari* table is variable among the different universities. The worst in term of tuples completeness is the Politecnico di Milano that has about 27% of incomplete tuples in the *aggiudicatari* table, this is the result of the incompleteness of *identificativoFiscaleEstero* attribute.

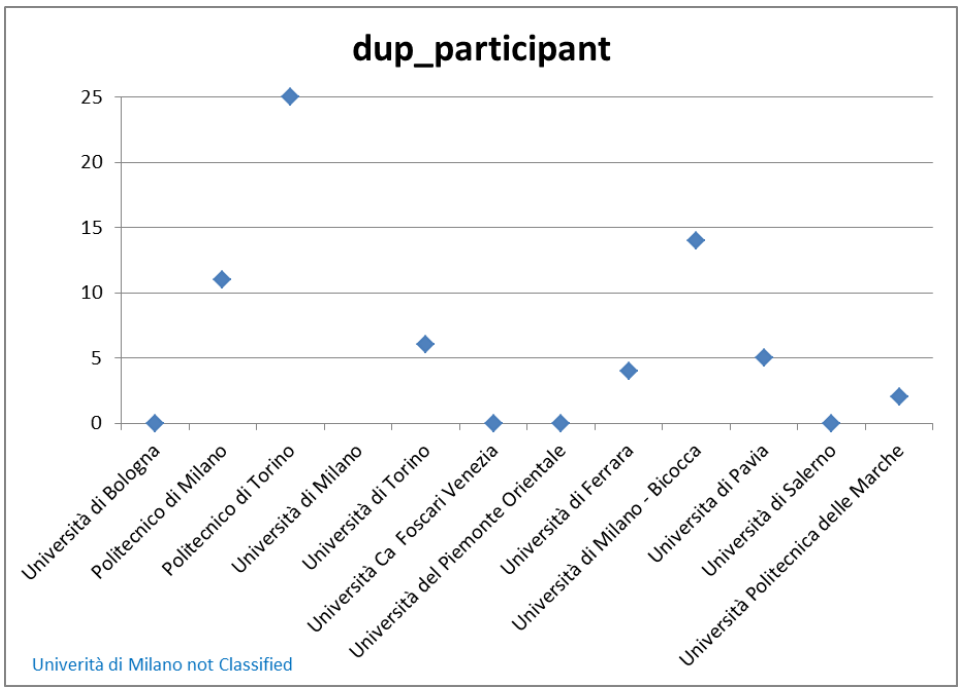



**Figure 58:**Participants duplicated

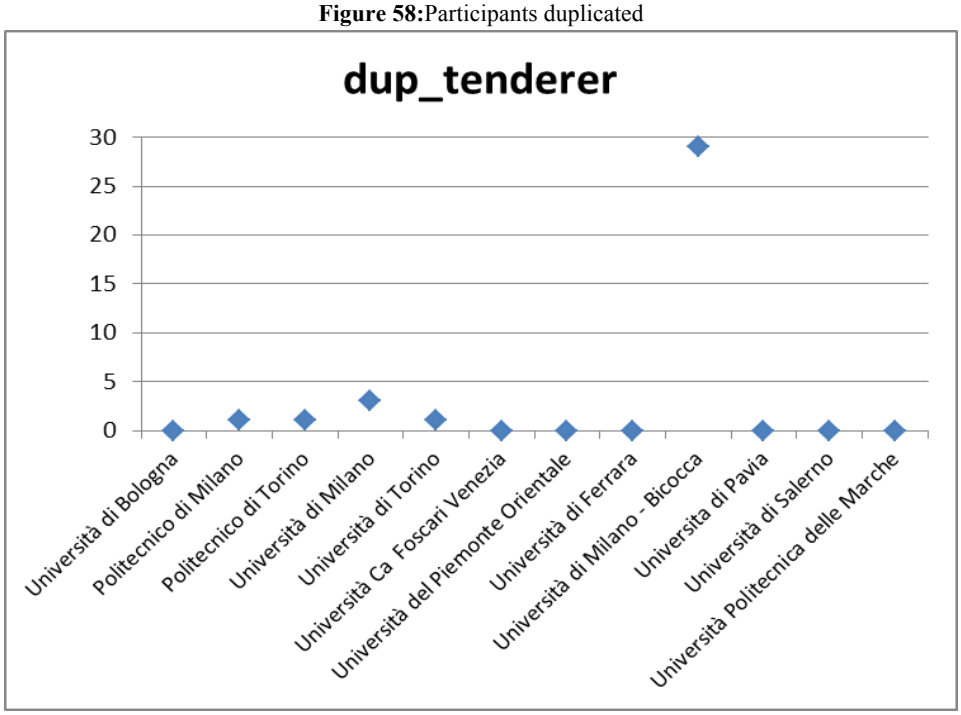



**Figure 59:**Successful tenderers duplicated

The figures 58 shows the total number of participants that are duplicated. We evaluate the number of duplicated participants because a participant can participate only once to a contract. For the Politecnico di Torino there are 25 participants which participate more than once to a contract. The number of duplicated participant should be equal to zero but in our case we have that 7 of the 12 analysed universities have duplicated participant in their contracts. A successful tenderer must appear only once in a contract, that is the same participant cannot win twice the contract. As shown in

figure 59, 4 of the 12 analysed universities have duplicated successful tenderer, the worst case is given by the University of Milano-Bicocca that has 29 duplicated successful tenderers.

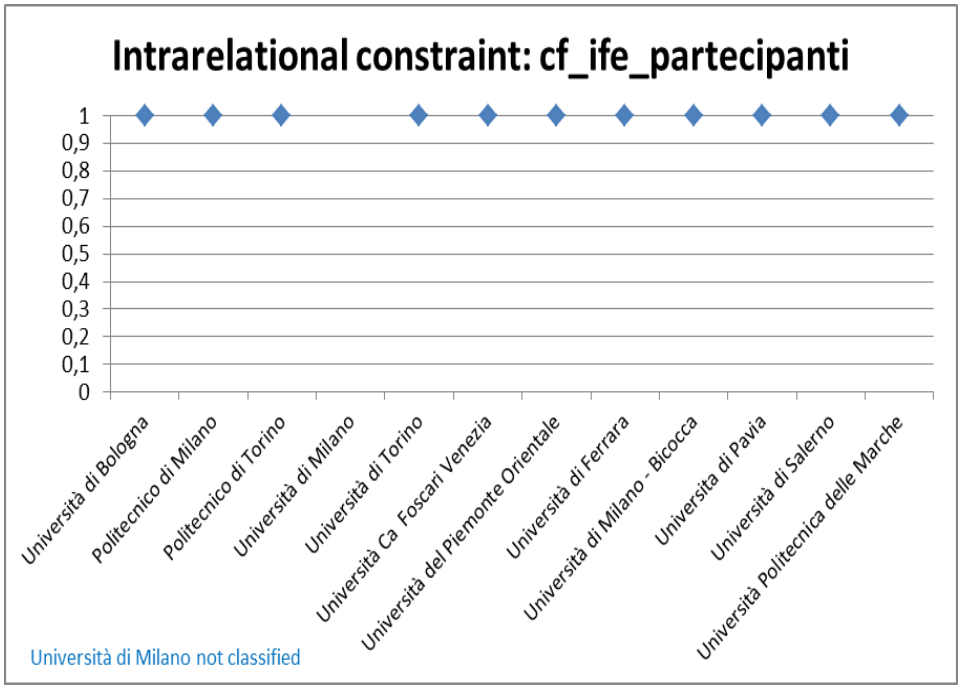



**Figure 60:**Intrarelational constraint cf_ife_partecipanti

The figure 60 shows that the 0% of tuples of the *partecipanti* table have simultaneously *codiceFiscale* and *identificativoFiscaleEstero* different by *null*. This result indicates that in the analysed files never occur that a participant has the presence of both the *codiceFiscale* and the *identificativoFiscaleEstero* elements. It is important to notice that University of Milano is not classified, since it does not provide any information of participants it is not possible to compute the metric.

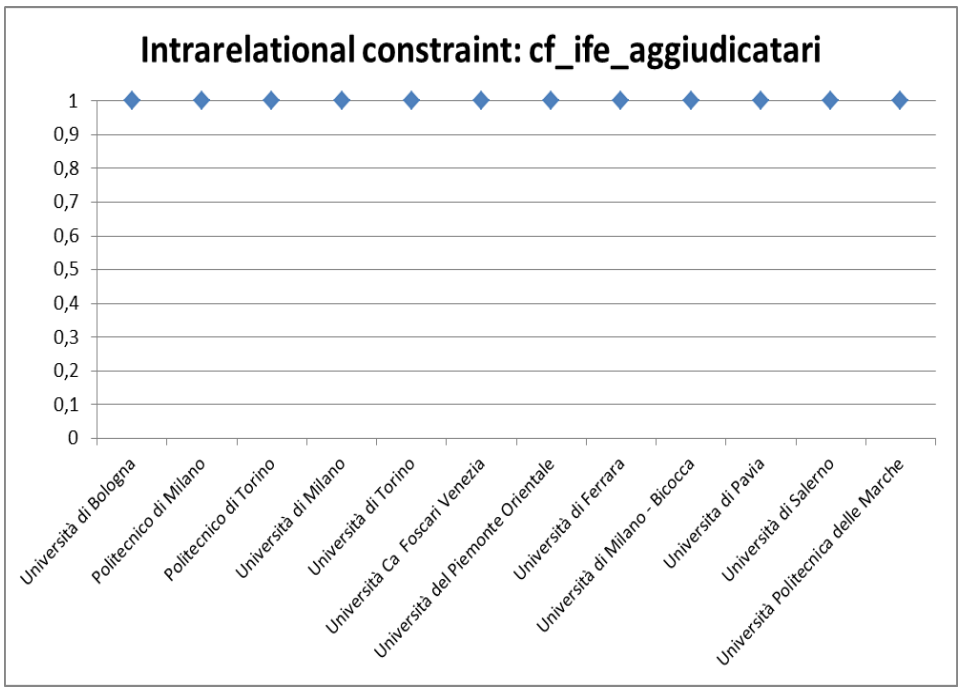



**Figure 61:**Intrarelational constraint cf_ife_aggiudicatari

Figure 61 shows that, even for the *aggiudicatari* table, the percentage of tuples that have simultaneously *codiceFiscale* and *identificativoFiscaleEstero* different by null, is 0.

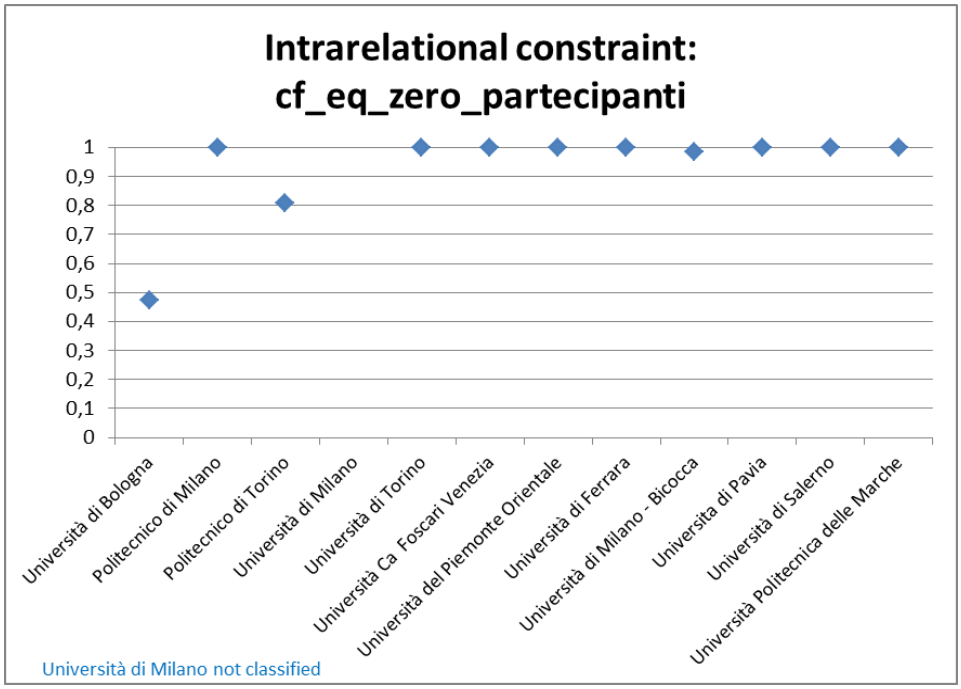



**Figure 62:**Intrarelational constraint cf_eq_zero_partecipanti

The University of Bologna and the Politecnico di Torino have respectively about the 53% and 20% of tuples of *partecipanti* table where *codiceFiscale* is equal to either 00000000000 or 0000000000000000, that is, values that do not allow to uniquely

identify a participant. It is not possible to compute the metric for the University of Milano that is not classified.

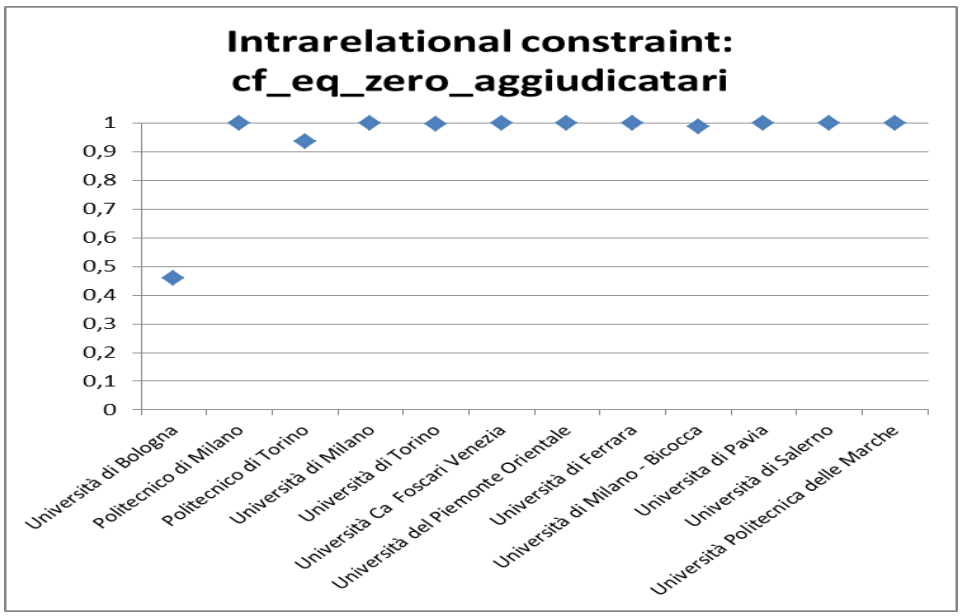



**Figure 63:**Intrarelational constraint cf_eq_zero_partecipanti

Figure 63 shows that the 55% of tuples of aggiudicatari table have the value of *codiceFiscale* equal to either 00000000000 or 0000000000000000. Looking at this result we can say that more than half of the successful tenderers cannot be uniquely identified and it is not possible to retrieve information about them from the source files.

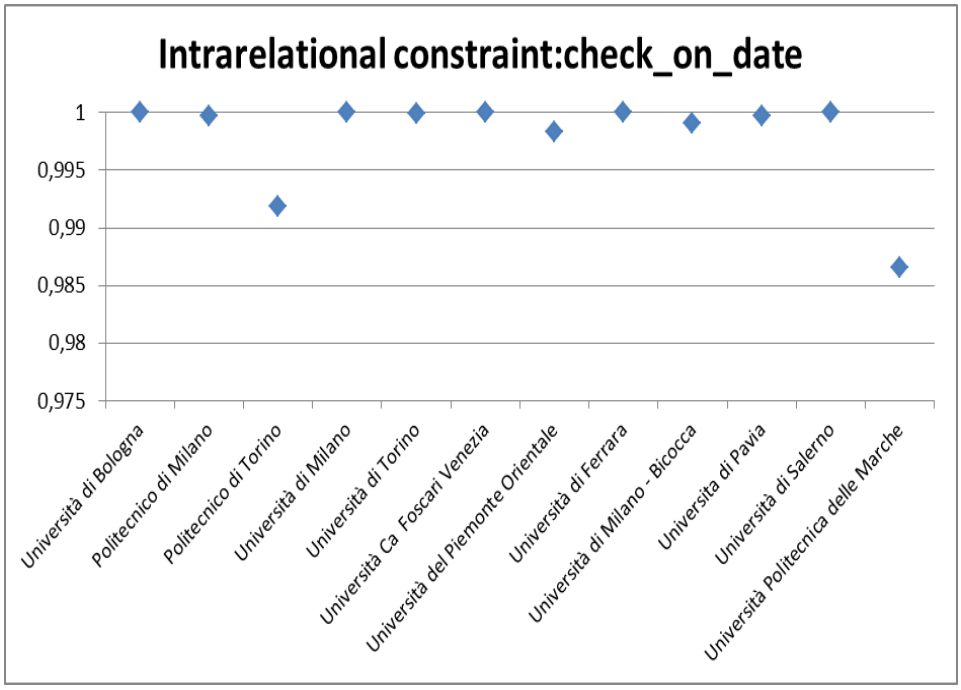



**Figure 64 :**Intrarelational constraint check_on date

The check_on_date metric returns a percentage that is about 100% for all the universities. This indicates that the *dataInizio* attribute is less recent than *dataUltimazione* in almost all lots provided by the different universities.

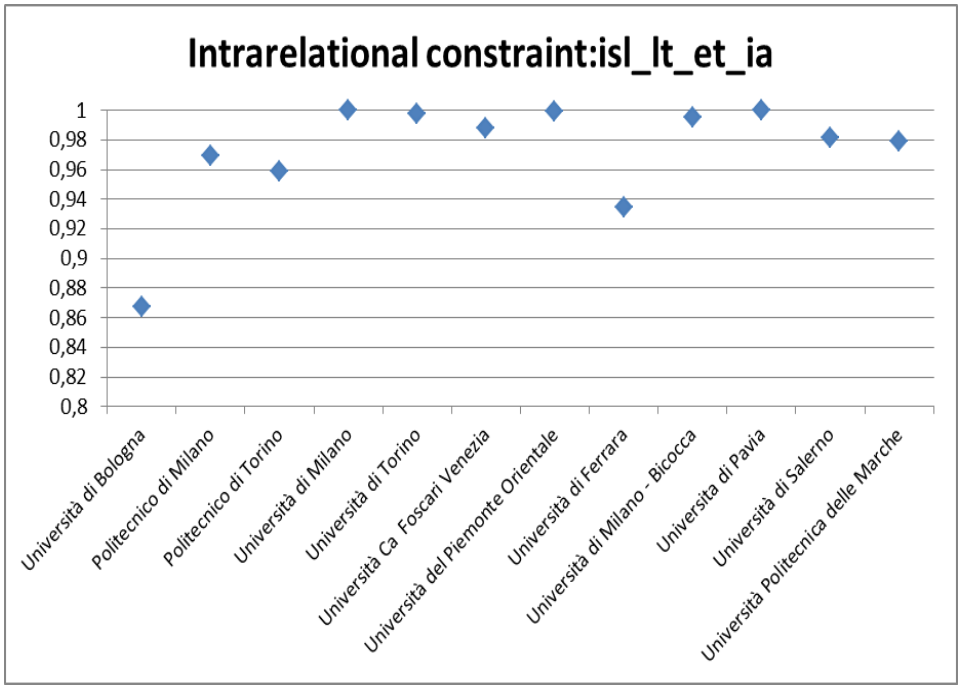



**Figure 65:**Intrarelational constraint isl_lt_et_ia

This metric highlights a very important aspect, that is, the percentage of contracts that have amount paid less than or equal to the award amount. Given its importance, it is desirable that the percentage is always the maximum or rather that there are no lots for which the sum of the amounts paid is greater than the award amount. From figure 65 we see that the University of Bologna has around the 14% of tuples in the *lotti* table in which the constraint is violated. This underlines that on the total of analysed lots, for 14% of the contracts were paid a sum greater than the award amount.

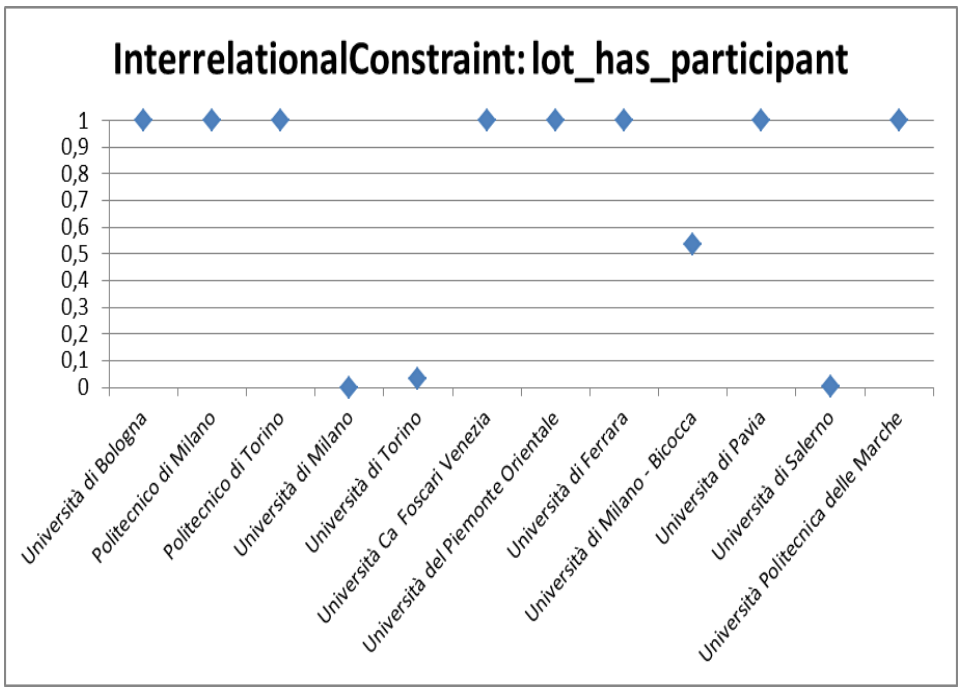



**Figure 66:**InterrelationalConstraint: lot_has_participant

The figure 66 shows that the percentage of lots which have successful tenderers and have participant is quite variable. The most interesting case in the University of Milano that have a percentage equal to zero. In the computation of the metrics of all the attributes in *partecipanti* table we have seen that the University of Milano is not classified because there aren't information on the participants in the source files. This metric shows clearly that in all contracts of the University of Milano are specified, for all lots the information about the successful tenderer but there is no information on participants. Although the University of Salerno has information about the participants, it has a very low percentage, this means that the information on the participants are specified only for some contracts.

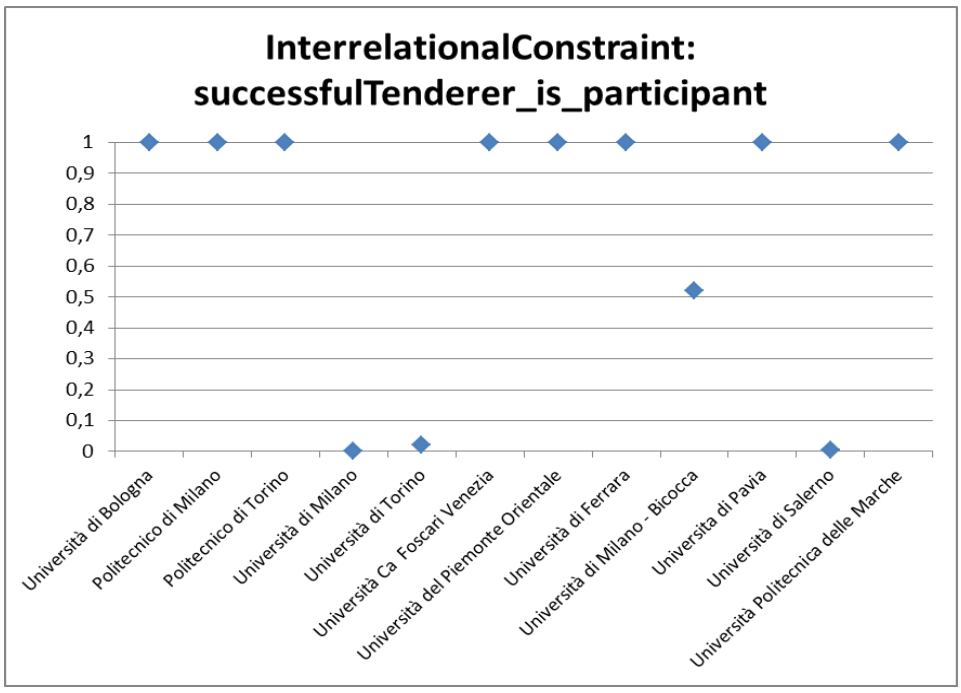



**Figure 67:**InterrelationalConstraint: successfulTenderer_is_participant

The percentage of lots in which the successful tenderer is a participant for the University of Milano is, obviously, equal to zero. This is a direct consequence of the total lack of information on participants that underlines that for the 100% of lots the successful tenderer does not appear in the list of the participants. We can notice that for Universities of Salerno the lot_has_participant and successfulTenderer_is_participant metrics return the same value. This makes us conclude that the successful tenderer is not participating in lots where it is not provided any information on the participants, while for lots in which participants are specified, the successful tenderer is always a participant.

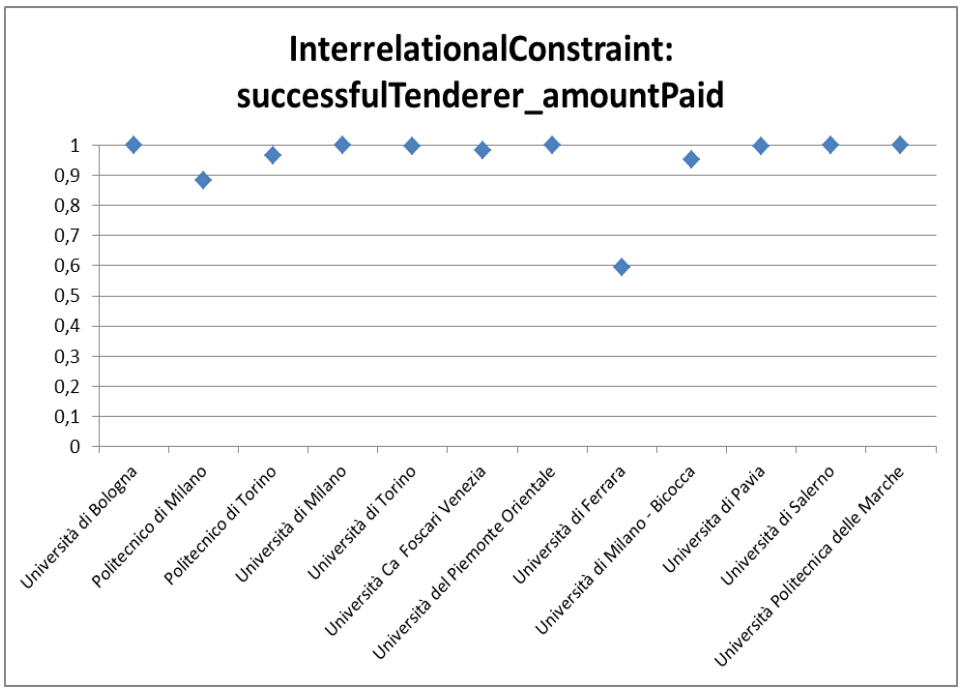



**Figure 68:**InterrelationalConstraint: successfulTenderer_amountPaid

The figure 68 shows the percentage of lots in which the successful tender is present and the amount paid is different by zero. This is another very important information because in many cases there aren't successful tenderers but is still delivered a sum of money. For the University of Ferrara there are around the 40% of contracts for which an amount of money is distributed but there aren't successful tenderers. Having no information on successful tenderers the money cannot be traced, that is, it is impossible to determine who receives the money.

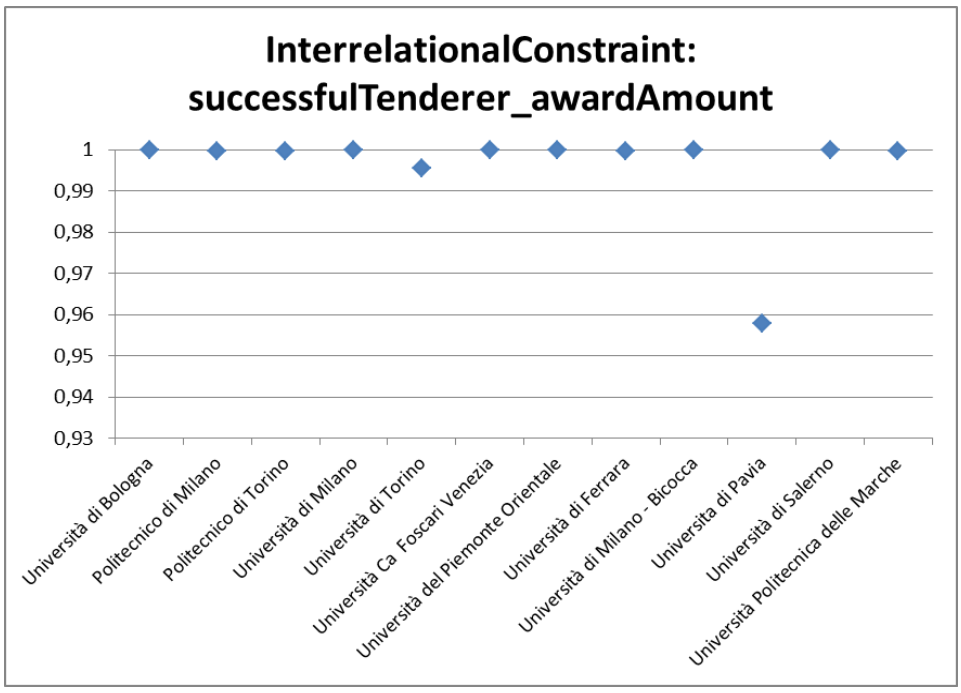



**Figure 69:**InterrelationalConstraint: successfulTenderer_amountPaid

The percentage of contracts in which the successful tenderer is present and the award amount is not zero is very high for the analysed cases. A very slight variation is given by the University of Pavia where about the 4% of contracts have successful tenderers but the award amount is zero. This case is less significant of the one described above because in the previous, public money are spent in a non-transparent and clear way while this metric only specifies the number of contracts that have a winner but with a total award of zero.

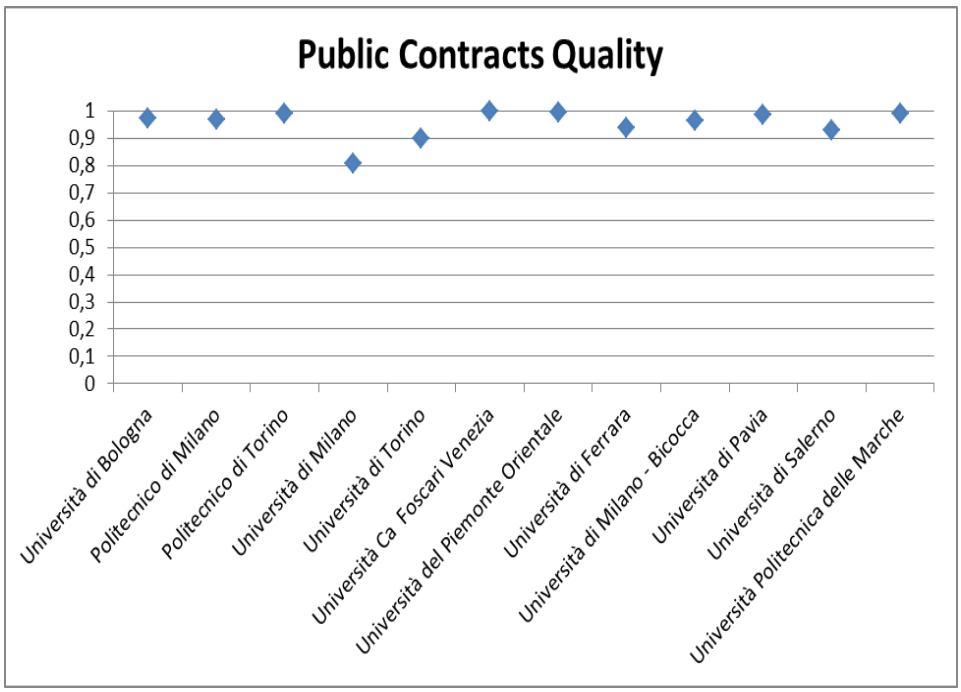



**Figure 70:**Public Contracts Quality

After the evaluation of all the metrics and the study of all the results we are able to provide the ranking of universities based on those that provide high quality data. We decided to aggregate the data through average function applied to all the percentages of correct cells obtained from the calculation of each metric for the single universities. The obtained results are shown in Figure 70. The University with the lowest average of correct cells is the University of Milano, while the University Ca Foscari of Venezia has the highest average of correct cells.

The final classification is:

| Position | University | Percentage |
|---|---|---|
| 1 | Università Ca Foscari Venezia | 99,91% |
| 2 | Università del Piemonte Orientale | 99,46% |
| 3 | Politecnico di Torino | 99,15% |
| 4 | Università Politecnica delle Marche | 99,10% |
| 5 | Universita di Pavia | 98,88% |
| 6 | Università di Bologna | 97,48% |
| 7 | Politecnico di Milano | 96,82% |
| 8 | Università di Milano - Bicocca | 96,79% |
| 9 | Università di Ferrara | 93,99% |
| 10 | Università di Salerno | 92,99% |
| 11 | Università di Torino | 90,08% |
| 12 | Università di Milano | 81,04% |

It can be said that the quality of data provided by the analysed Italian Universities is generally quite high. Although the quality is high, there are some unacceptable mistakes that should be avoided at the time of compilation of the summary tables. Error examples are the lack of information regarding the choice of the contractor (*sceltaContraente* element) of the University of Milano and the total lack of information on participants. For these errors, the University of Milano has the last place in the classification. Another mistake made by the University of Torino is to not provide, in many cases, the information of the participants. Such lack of information may be rated as less important than information that enable Authorities to check any illegality in the assignment of the contracts as the choice of the contractor. Another error that must be absolutely avoided is the one computed with the

successfulTenderer_amountPaid metric on which the University of Ferrara is evaluated as the worst one. Although the University of Milano is at the last position of the ranking, it should be highlight that the number of items published by this University is much greater than that provided by the University Ca Foscari of Venezia, which occupies the first place. An interesting comparison is the one between the Politecnico di Torino and the University Politecnica delle Marche, for the same percentage of correct cells Politecnico di Torino published about double lots published by the University Politecnica delle Marche.

In the following it is provided a summary table of all errors found in the analysed files and that we were able to capture through the metrics of the proposed model.

| Element | | Accuracy | Completeness | dup_participant | dup_tenderer | cf_ife_partecipanti | cf_ife_aggiudicatari | cf_eq_zero_partecipanti | cf_eq_zero_aggiudicatari | Check_on_date | isl_lt_et_ia | lot_has_participant | successfulTenderer_is_participant | successfulTenderer_is_amountPaid | successfulTenderer_awardAmount |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| lotti.cig | Out of Domain | X | | | | | | | | | | | | | |
| | Blank | X | | | | | | | | | | | | | |
| | Not Present | | X | | | | | | | | | | | | |
| lotti.codiceFiscaleProponente | Out of Domain | X | | | | | | | | | | | | | |
| | Blank | X | | | | | | | | | | | | | |
| | Not Present | | X | | | | | | | | | | | | |
| lotti.denominazione | Out of Domain | X | | | | | | | | | | | | | |
| | Blank | | X | | | | | | | | | | | | |
| | Not Present | | X | | | | | | | | | | | | |
| lotti.oggetto | Out of Domain | X | | | | | | | | | | | | | |
| | Blank | | X | | | | | | | | | | | | |
| | Not Present | | X | | | | | | | | | | | | |
| lotti.sceltaContraente | Out of Domain | X | | | | | | | | | | | | | |
| | Blank | X | | | | | | | | | | | | | |
| | Not Present | | X | | | | | | | | | | | | |
| lotti.importoAggiudicazione | Out of Domain | X | | | | | | | | | | | | | |
| | Blank | X | | | | | | | | | | | | | |
| | Not Present | | X | | | | | | | | | | | | |
| lotti.importoSommeLiquidate | Out of Domain | X | | | | | | | | | | | | | |
| | Blank | X | | | | | | | | | | | | | |
| | Not Present | | X | | | | | | | | | | | | |
| lotti.dataInizio | Out of Domain | X | | | | | | | | | | | | | |
| | Blank | X | | | | | | | | | | | | | |
| | Not Present | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| lotti.dataUltimazione | Out of Domain | X | | | | | | | | | | | | |
| | Blank | X | | | | | | | | | | | | |
| | Not Present | | | | | | | | | | | | | |
| partecipanti.codiceFiscale | Out of Domain | X | | | | | | | | | | | | |
| | Blank | X | | | | | | | | | | | | |
| | Not Present | | X | | | | | | | | | | | |
| partecipanti.identificativoFiscaleEstero | Out of Domain | | | | | | | | | | | | | |
| | Blank | | X | | | | | | | | | | | |
| | Not Present | | X | | | | | | | | | | | |
| partecipanti.ragioneSociale | Out of Domain | X | | | | | | | | | | | | |
| | Blank | | X | | | | | | | | | | | |
| | Not Present | | X | | | | | | | | | | | |
| partecipanti.ruolo | Out of Domain | X | | | | | | | | | | | | |
| | Blank | X | | | | | | | | | | | | |
| | Not Present | | X | | | | | | | | | | | |
| aggiudicatati.codiceFiscale | Out of Domain | X | | | | | | | | | | | | |
| | Blank | X | | | | | | | | | | | | |
| | Not Present | | X | | | | | | | | | | | |
| aggiudicatari.identificativoFiscaleEstero | Out of Domain | | | | | | | | | | | | | |
| | Blank | | X | | | | | | | | | | | |
| | Not Present | | X | | | | | | | | | | | |
| aggiudicatari.ragioneSociale | Out of Domain | X | | | | | | | | | | | | |
| | Blank | | X | | | | | | | | | | | |
| | Not Present | | X | | | | | | | | | | | |
| aggiudicatari.ruolo | Out of Domain | X | | | | | | | | | | | | |
| | Blank | X | | | | | | | | | | | | |
| | Not Present | | X | | | | | | | | | | | |
| Duplicated Participant | | | | X | | | | | | | | | | |
| Duplicated Successful Tenderer | | | | | X | | | | | | | | | |
| Participants codiceFiscale and IdentificativoFiscaleEstero both present | | | | | | X | | | | | | | | |
| Successful tenderer codiceFiscale and IdentificativoFiscaleEstero both present | | | | | | | X | | | | | | | |
| Participant codiceFiscale equal to zero | | | | | | | | X | | | | | | |
| Successful tenderer codiceFiscale equal to zero | | | | | | | | | X | | | | | |
| dataUltimazione less recent than dataInizio | | | | | | | | | | X | | | | |
| importoSommeLiquidate > importoAggiudicazione | | | | | | | | | | | X | | | |
| lots with successful tenderer without participants | | | | | | | | | | | | X | | |
| lots with successful tenders which is not a participant | | | | | | | | | | | | | X | |
| lots with successful tenderer not present and importoSommeLiquidate different by zero | | | | | | | | | | | | | | X |
| lots with successful tenderer present and importoAggiudicazione equal to zero | | | | | | | | | | | | | | X |

**Figure 71:** Errors and metrics used to detect them

*Chapter 7*

# 7. Conclusion

## 7.1 Synthesis of the work

The main objectives of this work were:

1. Extraction transformation and loading of data related to public contracts.
2. Definition of a set of metrics to create a framework to evaluate the quality of the data.
3. Using metrics defined by the framework to test its effectiveness and to conduct a competition on the quality of the data produced and published by several Italian Universities.

To develop the point 1. the extraction, transformation and loading approach have been used. At this stage it was important to design a database to store data relating to public contracts. This database is used as a starting point to carry out the analysis, for this reason we have defined transformations to be performed on the data before storing them in the database (Chapter 3). Obviously before the extraction of the data, a choice on them was made. The choice was public contracts published by Italian Universities. To achieve the objective 2. we introduced different model and the metrics used for each of them. After an analysis of the data to be evaluated we have defined a set of metrics, by introducing an hybrid model specially modelled for the evaluation of the public contracts (Chapter 4). Through automation of the metrics defined in the model it was possible to analyse large amounts of data, achieving the objective 3. The results obtained evaluating the quality of the data downloaded from the corporate website of the different Italian Universities, emphasizing certain aspects that can be improved as the accuracy and completeness rather than the consistency. Due to the problems identified has been possible to identify the universities that provide better quality data than the others (Chapter 6).

As a conclusion of the work, is important to point out that the advantage of the application developed is that it can be used to analyse any summary table of any government since, the structure of these tables is standard and is defined by the Authority for the supervision of public contracts. We have decided to use the data for

universities to test the effectiveness of the framework, but it can be used for data evaluation of summary tables on public contracts of any public administration.

## 7.2 Future Development

To build the framework we used two kinds of metrics, metrics that can be applied on any type of data and metrics which strictly depend on the domain, that is, public contracts. To define the second type of metric, for some of them we have been helped by the information contained in the XML schema while for others we looked at the data and we have defined some logical constraints on them. Future work would be to increase the level of analysis. Through detailed knowledge on the range of the award amounts that can be paid for each of the different choices of the contractor it is possible to check whether the contracts were entrusted all legally. This type of analysis requires specialized knowledge in the field of public contracts and requires the help of people with specific skills. A further development in the future would be to use the framework for evaluating the data provided on the website and those published by the Authority for the supervision of public contracts relating to the same public administration and check for any possible inconsistencies.

Bibliography:

Vademecum – Come rendere aperti I dati delle pubbliche Amministrazione : http://www.funzionepubblica.gov.it/media/982175/vademecumopendata.pdf.

Specifiche tecniche per la pubblicazione dei dati ai sensi dell'art.1 comma 32 legge n.190/2012.

Data Quality – Concept, Methodologies and Techniques. Batini & Scannapieca.

Methodologies for Data Quality Assessment and Improvement. ACM Computing Surveys, *41*(16). Batini, C., & Cappiello , C. (2009, luglio).

Beyond Accuracy: What Data Quality Means to Data Consumers. Journal of Management Information Systems. Wang R. Y., & Strong D. M. (1996).

Valutazione delle qualità degli Open Data - Tesi di laurea Magistrale. Torino: Politecnico di Torino. Orozco,Torchiano , & Vetrò (2013).

La qualità degli Open Data in Italia: il caso di Open Coesione – Tesi di laurea Magistrale. Torino: Politecnico di Torino, L.Canova (2014).

The Open Definition:
www.opendefinition.org

Open Data:
www.wikipedia.it

Open Data Institute:
www.theodi.org

Open Data and the Future of Civic Innovation:
www.beyondtransparency.org

A brief history of open data:
www.paristechreview.com/2013/03/29/brief-history-open-data

The 8 principles of open government data:
www.opengovdata.org

The Open Data Index:
http://index.okfn.org

The Open Data Index:
http://index.okfn.org/place

The Open Data Index:
http://index.okfn.org/place/2013

What is an XML Schema?:
www.msdn.microsoft.com/en-us/library/ms765537(v=vs.85).aspx

Java Architecture for XML Binding:
http://www.oracle.com/technetwork/articles/javase/index-140168.html

JDBC Database Access:
https://docs.oracle.com/javase/tutorial/jdbc/

JDBC Database Access:
https://docs.oracle.com/javase/tutorial/jdbc/overview/index.html

Five Star Open Data. Berners-Lee:
http://5stardata.info/en/