Data Quality Evaluation of Scientific Datasets
A Case Study in a Policy Support Context

Antonella Zanzi[1,2] and Alberto Trombetta[1]
[1]*Dipartimento di Scienze Teoriche e Applicate, Università degli Studi dell'Insubria, via Mazzini 5, 21100 Varese, Italy*
[2]*Joint Research Centre, European Commission, via Enrico Fermi 2749, 21027 Ispra, Italy*

Keywords: Data Quality, Data Constraints, Data Dependencies, Order Dependencies, Existence Constraints.

Abstract: In this work we present the rule-based approach used to evaluate the quality of scientific datasets in a policy support context. The used case study refers to real datasets in a context where low data quality limits the accuracy of the analysis results and, consequently, the significance of the provided policy advice. The applied solution consists in the identification of types of constraints that can be useful as data quality rules and in the development of a software tool to evaluate a dataset on the basis of a set of rules expressed in the XML markup language. As rule types we selected some types of data constraints and dependencies already proposed in data quality works, but we experimented also the use of order dependencies and existence constraints. The case study was used to develop and test the adopted solution, which is anyway generally applicable to other contexts.

1 INTRODUCTION

Awareness of the critical importance of data quality issues has grown rapidly in the last decades in scientific, industrial and governmental contexts. For new archives or data collections the prevention is obviously a fundamental aspect, still, issues can arise in the data management process, when data are derived from other data, and when a dataset results from the integration of multiple data sources. Moreover, in some cases, it is necessary to use datasets received from external unverified sources without having any control on the original acquisition process.

An approach is to attempt to cleanse data. Data cleaning is not a simple task, it is highly context dependent, and, in many cases, data can be cleaned effectively only with some human intervention, since fully automated cleaning procedures could lead to a loss of useful information.

The discovery of incorrect data, the first necessary step in data cleaning, is – in most cases – challenging; moreover, when the presence of errors is recognized it is not always feasible to trace back the correct values (e.g., detecting inconsistencies among data may not be sufficient to determine which record is at fault) or it is not possible to correct a dataset (e.g., changes in the original dataset are not allowed).

Another relevant aspect related to the quality of a dataset is its fitness for the intended purpose, which is one of the definition provided for data quality, as for example in (Juran, 1964) "Data are of high quality if they are fit for their intended use in operations, decision-making, and planning" and in (Shanks and Corbitt, 1999), where the authors adopted the definition of quality as "fitness for purpose". It may occur that a dataset containing correct data is not useful in a specific context, for example because a different level of detail is requested, or for insufficient data coverage. In these cases, efforts need to be devoted in assessing the quality level of datasets in order to evaluate their fitness for the intended purpose.

In the present work, we focus on the evaluation of the quality of scientific datasets[1] considering mainly the accuracy, consistency and coverage data characteristics and we use, as case study, some datasets collected by the Joint Research Centre of the European Commission, the Commission's in-house science service having the mission to provide European Union (EU) policies with independent, scientific and technical support. The adopted solution consists in the identification of types of constraints that can be useful as data quality rules and in the development of a software tool to evaluate a dataset on the basis of a specified set of rules.

[1]For a discussion about scientific dataset definitions refer to (Renear et al., 2010).

Zanzi A. and Trombetta A..
Data Quality Evaluation of Scientific Datasets - A Case Study in a Policy Support Context.
DOI: 10.5220/0004476401670174
In *Proceedings of the 2nd International Conference on Data Technologies and Applications* (DATA-2013), pages 167-174
ISBN: 978-989-8565-67-9

As constraints and their enforcement play a key role in the maintenance of data integrity into a database, rules and their verification can play a key role in the evaluation and assessment of the consistency of a dataset. The consistency of a dataset can be defined in terms of constraints, and inconsistencies in the dataset can be detected as violations of these constraints. Classic integrity constraints are relevant for data quality and for data cleaning, however, they do not capture all the data quality concerns. For this reason, new forms of quality constraints, which can be considered an extension of usual semantic integrity constraints in databases, are proposed and investigated. Moreover, data quality rules can play a further role: they can help in verifying the suitability of a dataset to be used for a certain purpose.

As data quality rule types to be used in our tool, first of all, we selected some types of data constraints and dependencies already proposed in data quality works, as for example association rules, Functional Dependencies (FDs) and Conditional Functional Dependencies (CFDs), the latest being FDs holding on a subset of the relation instance and recently introduced in the data cleaning context (Bohannon et al., 2007). In addition, we considered *order dependencies* (Ginsburg and Hull, 1983) and *existence constraints* (Atzeni and Morfuni, 1986) because they can be useful in the data quality evaluation context, even if, to our knowledge, they have not yet been used in the data quality context.

2 THE CASE STUDY CONTEXT

In the European Union the fisheries sector is managed through the Common Fisheries Policy (CFP). Since the CFP was first established in 1983, scientific advice has increasingly become a major part of the fisheries management decision-making process.

The are several impediments to the rational control of marine resources, and one of them is inadequate data. Fisheries management decisions are often based on population models, but the models need data to be accurate (Chen, 2003).

In order to allow a pan-European set of data to be used for policy advice, the Commission Regulation (EC) No. 665/2008 (European Commission, 2008) established the Data Collection Framework (DCF), a Community framework for the collection of data in the fisheries sector. The framework requires Member States to collect biological and economic data of many European fisheries and related fisheries sectors, and to provide access to these data for fisheries management advice, scientific publication, public debate and stakeholder participation in policy development. The collected fisheries data can be divided in the following datasets:

- Economic Data: employment, expenditure, capital and investments, fishing rights, and direct subsidies;
- Biological Data: length and age distribution, maturity data by age and length, sex ratio by age and length, discards, and discards length distribution;
- Effort Data: days at sea and energy consumption;
- Transversal Data: capacity, landings, and prices;
- Data from Surveys (i.e., sampling at sea).

Data are normally aggregated at various levels, for example by year, by area, by fleet segment, by mesh size, and special management conditions. Details about the datasets, which are not reported here for lack of space, can be found in a dedicated Web site (https://datacollection.jrc.ec.europa.eu) where the collected data are published. The Joint Research Centre (JRC) on behalf of the Directorate-General for Maritime Affairs and Fisheries of the European Commission, collects, checks and maintains the fisheries data reported by EU Member States in the framework of the DCF. The data quality checks performed at JRC is concretely helping Member States in assessing the quality of the data they provide and also in improving the quality of their data management process. Detailed information about the activities carried out by JRC in the context of the DCF can be found in the already mentioned Web site.

To illustrate the processes involved in the fisheries data submission from Member States, with particular emphasis on the activities addressing data quality verification and improvement, we use an Information Product Map (IP-MAP)[2], which is a graphical model for the description of information production processes (Shankaranarayan et al., 2000). The purpose of the IP-MAP shown in figure 1 is to position the step of the evaluation of data quality rules (highlighted in gray in the diagram) in the case study data management cycle. During the data submission procedure several preliminary checks are carried out, while other checks are performed after the submission procedure is concluded and the received data are stored in the staging database. The following steps related to data quality aspects are shown in the diagram:

- Data Domain Check - Each value is checked against the assigned domain.
- Duplicate Detection - Duplicate records and records referring to the same entities (e.g., the

[2]The list of the constructs that can be used to build IP-MAPs can be found in the appendix.

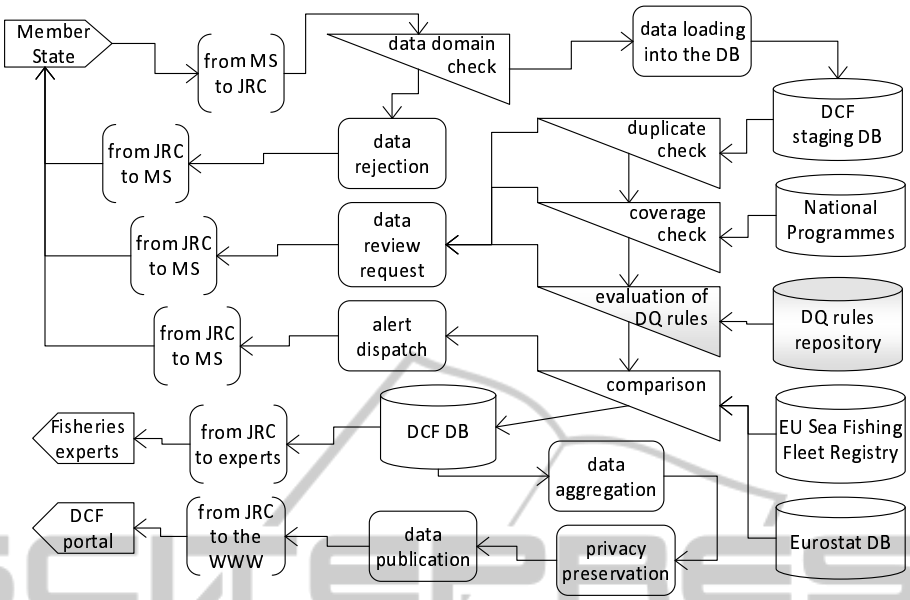

Figure 1: An IP-MAP for the case study.

same fleet segment) but with different values are identified.

- Evaluation of Data Quality Rules - In the diagram, the data quality block labeled *evaluation of DQ rules*, refers to the step of data quality evaluation based on a set of rules mainly provided by experts in fisheries data.
- Coverage Check - For each country, coverage checks are carried out to ensure that all the required data for each fleet segment has been submitted. To establish how many and which fleet segments should be reported by each country, the National Programmes of each Member State are consulted.
- Comparison with other Data Sources - Data from the EU Sea Fishing Fleet Register and from the Eurostat's archives are used to assess the consistencies of some of the provided data aggregated at national level.

3 DATA QUALITY RULES AND THEIR EVALUATION

We addressed two main targets: the expression of quality concerns and their evaluation against concrete instances. We adopted the solution to specify the rules using the XML markup language (the DTD for the used XML structure is shown in the appendix) and we developed a tool to evaluate a dataset against a specified set of rules. The developed tool, which is Java technology based, provides functionalities to express data quality rules and to evaluate them against a dataset stored in a relational database, identifying the data subsets that do not comply with the defined rules. In order to facilitate the input of the rules, predefined templates were built for each type of rule accepted by the tool; after a template has been completed with the details of a rule, the correct XML code for that rule is automatically generated. To validate the XML format, in addition to DTD and XML Schema, we used Schematron (van der Vlist, 2007), which is a structural based validation language. The data to be checked are stored in a relational database; in particular, for the case study we used PostgreSQL. In the rules, it is necessary to specify the tables containing the data; instead of tables, views can be used as well, allowing more flexibility in the definition of data constraints. For each defined rule, the tool identifies all the records that do not comply with it and stores them in views or tables inside the same database containing the source dataset.

3.1 Types of Rules

The rules managed by the developed tool are classified in the following types:

Functional Dependency Rule. Functional dependencies among attributes of the same relation can be defined with this type of rule, including CFDs (i.e., FDs holding on a subset of the relation instance) and CFD^{p}s, the latest being CFDs specifying patterns of data values with predicates ($\neq$, $<$, $>$, $\leq$, $\geq$) (Chen et al., 2009). The

left-hand side and the right-hand side of the rule can contain one or more attributes. Moreover, it is possible to add conditions connected by means of boolean operators in order to express CFDs and CFD^{p}s (in the XML rule format, conditions are recorded using the tag `when`).

Conditional Constraint Rule. This type of rule can specify a constraint that has to be valid if a given condition is satisfied and can be used to define any constraint that can be written in the form `if-then` among attributes of the same relation. Association rules, constant CFDs and constant CFD^{p}s (namely, CFDs and CFD^{p}s with constant patterns only) can be expressed with this kind of rule. The `if` and `then` components of the rule can contain one or more conditions, connected by means of boolean operators, on attributes of the same table.

Differential Dependency Rule. For this rule we refer to a new kind of dependency called *differential dependency*. It was proposed in (Song and Chen, 2011) and it can be used to specify constraints on distances over attributes of the same relation. In the developed tool, we limited the application of differential dependencies only to numerical attributes, but we introduced the possibility to apply these dependencies on non-overlapping sets of tuples on the basis of specified attributes (which in the XML format are expressed using the tag `partition_on`).

Check Constraint Rule. A check constraint rule consists in boolean conditions, which can contain arithmetic operators, on single attributes or among attributes belonging to the same or to different relations. When more than one table is involved, it is required to specify the criteria to be used to join the tables themselves (in the XML rule format, joining criteria are labeled with the tag `join_on`).

Existence Constraint Rule. Referring to the definition of existence constraints introduced in works related to databases with incomplete information (Atzeni and Morfuni, 1986), we defined existence constraint rules among attributes belonging to the same or to different relations.

Order Dependency Rule. Adopting the order dependency definition provided in literature (Ginsburg and Hull, 1983), we extended it defining a rule type to express direct and inverse order constraints among attributes belonging to the same relation.

In the rest of the paragraph, we will give more details about the existence constraint and order dependency rules managed by the tool, providing examples of SQL statements that can be used to identify non-complying data in a dataset. Note that in the shown SQL statements, XML tags are referred using Xpath-like expressions (for the complete structure refer to the DTD shown in the appendix).

3.1.1 Existence Constraint Rules

Two types of existence constraints, respectively called existence constraints and disjunctive existence constraints, have been defined in works related to databases with incomplete information (Atzeni and Morfuni, 1986). In contrast with the definitions proposed in literature, the implemented rule manages only attributes instead of sets of attributes in the left-hand side of disjunctive existence constraints and in both sides of existence constraints.

More formally, given a relation schema $R(U)$ and a relation instance r over R, this type of rule can express dependency-like and disjunctive-like existence constraints defined as follows:

- A dependency-like existence constraint $A \rightarrow B$ (read A requires B), where $A, B \in U$, holds over r if, for each tuple $t \in r$, $t[A] \neq$ null implies $t[B] \neq null$.
- A disjunctive-like existence constraint $A \rightarrow S$, where $A \in U$ and $S = \{Y_1, \ldots, Y_n\}$ is a set of sets of attributes (with $Y_1, \ldots, Y_n \subseteq U$), holds over r if, for each tuple $t \in r$, $t[A] \neq null$ then $\exists\, Y_i \in S$ (with $1 \leq i \leq n$) such that $\forall\, B \in Y_i\ t[B] \neq null$.

In addition, this kind of rule allows the definition of a *not null* constraint on a single attribute and a bidirectional existence constraint between a pair of attributes (if one of the two attributes exists in the dataset, the second has to exist as well) contained in the same table or in different tables.

The scope of the rule can be limited to a subset of the relation instance through conditions (which can be expressed in the XML format using the tag `when`). In addition, when two tables are referred, it is required to specify the criteria to be used to join them (in the XML rule format, joining criteria are labeled with the tag `join_on`).

General SQL statements to retrieve non-complying records for dependency-like, disjunctive-like and bidirectional existence constraints, when the involved attributes belong to different tables, are listed in the following:

```
SELECT //rule_ec//column_name
FROM //lhs_ec//table_name
LEFT OUTER JOIN //rhs_ec//table_name
ON (//rule_ec/join_on)
WHERE //lhs_ec//column_name IS NOT NULL
AND //rhs_ec//column_name IS NULL
AND //rule_ec/when
```

```
SELECT //rule_ec//column_name
FROM //lhs_ec//table_name
LEFT OUTER JOIN //rhs_ec/table_name
ON (//rule_ec/join_on)
WHERE //lhs_ec//column_name IS NOT NULL
AND
(//rhs_ec/disj_attr[i]/column_name[1] IS NULL
OR
//rhs_ec/disj_attr[i]/column_name[k] IS NULL)
AND //rule_ec/when

SELECT //rule_ec//column_name
FROM //rule_ec/column[1]/table_name
FULL OUTER JOIN //rule_ec/column[2]/table_name
ON (//rule_ec/join_on)
WHERE (//rule_ec/column[1]/column_name IS NULL
AND
//rule_ec/column[2]/column_name IS NOT NULL)
OR (//rule_ec/column[2]/column_name IS NULL
AND
//rule_ec/column[1]/column_name IS NOT NULL)
AND //rule_ec/when
```

In the case study, this kind of rule was used to express constraints like the following one: "For every active fleet segment, landing data (weight and value) have to be provided".

3.1.2 Order Dependency Rules

This kind of rule can be used to define order dependencies among attributes: given a relation schema $R(U)$, an order dependency $X \rightarrow Y$ $(X, Y \subseteq U)$ holds if an order over the values of each attribute in X implies an order over the values of each attributes of Y.

With this rule type it is possible to express different orderings for the attributes contained in the left-hand side and the right-hand side of the dependency, allowing the definition of both direct and inverse order dependencies.

More formally, given a relation schema $R(U)$ and a relation instance r over R, this type of rule can be used to define a dependency $X \rightarrow Y$ $(X, Y \subseteq U)$ such that:

- $X \rightarrow_{\preceq} Y$ denotes a *direct* order dependency if, for every pair of tuples s and $t \in r$, $s[X] \preceq t[X]$ implies $s[Y] \preceq t[Y]$, where $s[X] \preceq t[X]$ if $s[A] \leq t[A]$ $\forall A \in X$, and $s[Y] \preceq t[Y]$ if $s[B] \leq t[B]$ $\forall B \in Y$.
- $X \rightarrow_{\succeq} Y$ denotes an *inverse* order dependency if, for every pair of tuples s and $t \in r$, $s[X] \preceq t[X]$ implies $s[Y] \succeq t[Y]$, where $s[X] \preceq t[X]$ if $s[A] \leq t[A]$ $\forall A \in X$, and $s[Y] \succeq t[Y]$ if $s[B] \geq t[B]$ $\forall B \in Y$.

The scope of the rule can be limited to a subset of the relation instance through conditions (which can be expressed in the XML format using the tag `when`). In addition, it is possible to apply the rule on non-overlapping sets of tuples on the basis of specified attributes (which in the XML format are expressed using the tag `partition_on`).

The following SQL statement[3] identifies the records that do not satisfy a *direct* order dependency rule and provides, for each selected record, the number of failed comparisons.

```
WITH tmpView AS (
 SELECT DISTINCT //lhs/column_name,
 //rhs/column_name,
 //rule_od/partition_on/column_name as partCol
 FROM /rule_definition/table_name
 WHERE //rule_od/when
)
SELECT tableName.//lhs/column_name,
 tableName.//rhs/column_name, count(*)
FROM tmpView,
 /rule_definition/table_name as tableName
WHERE //rule_od/partition_on/column_name[j] =
      tmpView.partCol[j] AND //rule_od/when
AND ((tableName./lhs/column_name[i] <=
      tmpView.//lhs/column_name[i]
AND (NOT tableName.//rhs/column_name[1] <=
        tmpView.//rhs/column_name[1]
OR NOT tableName.//rhs/column_name[k] <=
      tmpView.//rhs/column_name[k]))
OR (tmpView.//lhs/column_name[i] <=
    tableName./lhs/column_name[i]
AND (NOT tmpView.//rhs/column_name[1] <=
        tableName.//rhs/column_name[1]
OR NOT tmpView.//rhs/column_name[k] <=
      tableName.//rhs/column_name[k])))
GROUP BY tableName.//lhs/column_name,
         tableName.//rhs/column_name
```

An example of order dependency used in the case study is the following one: "In a survey at sea, an inverse order dependency holds among registered temperatures and hauling depths".

3.2 The used Metric

In the case study context, in order to quantify the quality of the data provided by Member States, the main used metric is strictly connected with the adopted data quality rules; in fact, the used metric is based on the number of records satisfying the rules specified for the dataset.

Given a set of rules, and called respectively:

- N the total number of the rules
- w_r the weight (relevance) of the rule r
- S_r the number of records satisfying the rule r
- U_r the number of records not satisfying the rule r

$$I_{DQ_w} = \sum_{r=1}^{N} -\frac{I_r * w_r}{N} \qquad I_r = \frac{U_r}{S_r + U_r} \qquad (1)$$

[3]The SQL *WITH* clause implements the Common Table Expression defined in the SQL:1999 standard.

The values of I_{DQ_w} will be 0 or a negative number, where 0 is the best value for the index.

Due to the types of rules used in the present work, this kind of measure mainly refers to the accuracy, consistency and coverage data characteristics and, in the case study context, it was used to compare different datasets of different years, or subsets of the whole dataset (e.g., data by fleet segment or Member State).

4 RELATED WORKS

Data quality issues have been faced from different point of views and with several approaches.

Prevention approaches refer, for example, to proper database design and to the use of integrity constraints, but also to appropriate business process management avoiding the generation of low data quality. Common diagnostic approaches are database profiling and exploratory data analysis (Dasu and Johnson, 2003). Currently used corrective approaches, which are provided by the available data cleaning tools, comprise data cleaning methods for attributes (Hellerstein, 2008), duplicate detection techniques (Elmagarmid et al., 2007), and virtual repair methods (Bertossi and Chomicki, 2003). Some tools concentrate on a specific domain, such as cleaning names and address data, or on a specific cleaning phase, such as duplicate detection; at the contrary, in the Extraction Transformation Load (ETL) tools, which aim at helping in the construction of data warehouses, the provided built-in data cleaning capabilities cover a large part of the data transformation needs. A survey with a detailed feature comparison of data cleaning tools both of academic and industrial origin can be found in (Barateiro and Galhardas, 2005); while for an introduction to the issues faced and approaches used in ETL tools refer to (Vassiliadis, 2009).

Talking about *data quality rules* one can refer to rules used with different purposes. Integrity constraints are examples of rules normally enforced on databases and rules are used in data cleaning and ETL tools in order to express transformations to be applied to data.

A business rule management system (Bajec et al., 2000) is a software system used to define, execute and monitor the decision logic that is used by operational systems within an organization. The decision logic is represented through business rules meant to capture knowledge of constraints in an organization. The main focus of this kind of systems is not on data quality, however, they can contribute in improving the quality of the data produced by the business processes.

Some literature works proposed to use association rules extracted from a dataset as a means to discover dirty data; for example, in (Hipp et al., 2001) the authors present a rule-based data mining approach for outlier detection.

Recent works proposed to use FDs and extensions to FDs for data cleaning purposes in the context of relational databases. For example, in (Pivert and Prade, 2010) the authors consider the case where dirtiness corresponds to the violation of one or several FDs. The use of the recently proposed CFDs as a method for inconsistency detection and repairing is discussed in (Cong et al., 2007). In *Semandaq* (Fan et al., 2008), a tool using CFDs for data cleaning purposes, users can specify CFDs through the drag and drop functionality provided in the user interface. Another tool, called *Data Auditor*, is presented in (Golab et al., 2010) and supports more types of constraints (i.e., CFDs, *conditional inclusion dependencies* and *conditional sequential dependencies*) for testing data inconsistency and completeness. CFD^{p}s, which extend CFDs, were introduced, as already mentioned, in (Chen et al., 2009); in the same work, the authors describe an approach to automatically generate SQL queries to select all the data that violate a set of CFD^{p}s encoded in relational tables.

To the best of our knowledge, existence constraints and order dependencies have not yet been used in the context of data quality evaluation.

Existence constraints were introduced in works related to databases with incomplete information (Atzeni and Morfuni, 1986); moreover, in the database design context, the term existence dependency was used as a synonym of participation constraint (Elmasri and Navathe, 2000), a structural constraint indicating that the existence of an entity depends on its being related to another entity.

Order dependencies, instead, were introduced for the first time in the context of database systems by (Ginsburg and Hull, 1983), later *pointwise ordered functional dependencies* and *lexicographical ordered functional dependencies* were formally defined in (Ng, 1999). Moreover, in a recent work, order dependencies were used in the query optimization context (Szlichta et al., 2012).

Assessment of data quality can be performed in relation to several data quality characteristics and, for each of the selected characteristic, specific variables to be measured can be chosen; examples of metrics of this kind are shown in (Lee et al., 2006). The metric used in our work, instead, is more comparable with the types of metrics (also mentioned in (Lee et al., 2006)) developed to measure data adherence to integrity constraints in relational databases.

5 CONCLUSIONS

In order to deal with inconsistencies among data and to evaluate the data quality level of a dataset, we proposed an approach based on rules expressed with an XML-based syntax and we developed a tool to deploy such rules on a dataset. The proposed approach allows a user to easily define different kinds of rules in the same environment, without to deal with direct manipulation of XML trees. In fact, in order to express data quality rules, a user needs to know the database schema of the dataset and the types of rules managed by the tool. The developed tool manages a predefined number of rule types, but it can be easily extended in order to deal with other types of rules.

As further work, we seek to perform an assessment of the proposed approach on larger – and possibly different – datasets, in order to validate it on other applicative domains.

ACKNOWLEDGEMENTS

The authors acknowledge the contribution of the JRC colleagues involved in the DCF activities, while being solely responsible for possible incomplete or erroneous statements.

Disclaimer. The content of this work reflects only the opinion of the authors and may not be regarded as stating an official position of the European Commission.

REFERENCES

Atzeni, P. and Morfuni, N. (1986). Functional dependencies and constraints on null values in database relations. *Information and Control*, 70(1):1–31.

Bajec, M., Krisper, M., and Rupnik, R. (2000). Using business rules technologies to bridge the gap between business and business applications. In Rechnu, G., editor, *Proceedings of the IFIP World Computer Congress*, pages 77–85.

Barateiro, J. and Galhardas, H. (2005). A survey of data quality tools. *Datenbank-Spektrum*, 14/2005:15–21.

Bertossi, L. and Chomicki, J. (2003). Query answering in inconsistent databases. In Chomicki, J., Saake, G., and van der Meyden, R., editors, *Logics for Emerging Applications of Databases*, pages 43–83. Springer.

Bohannon, P., Fan, W., Geerts, F., Jia, X., and Kementsietsidis, A. (2007). Conditional functional dependencies for data cleaning. In *Proceedings of the Int'l Conference on Data Engineering*, pages 746–755.

Chen, W., Fan, W., and Ma, S. (2009). Analyses and validation of conditional dependencies with built-in predicates. In Bhowmick, S., Kung, J., and Wagner, R., editors, *Proceedings of the Int'l Conference on Database and Expert Systems Applications*, volume 5690 of *LNCS*, pages 576–591. Springer-Verlag.

Chen, Y. (2003). Quality of fisheries data and uncertainty in Stock Assessment. *Scientia Marina*, 67(Suppl. 1):75–87.

Cong, G., Fan, W., Geerts, F., Jia, X., and Ma, S. (2007). Improving data quality: Consistency and accuracy. In *Proceedings of the Int'l Conference on Very Large Data Bases*, pages 315–326. VLDB Endowment.

Dasu, T. and Johnson, T. (2003). *Exploratory Data Mining and Data Cleaning*. Wiley.

Elmagarmid, A., Ipeirotis, P., and Verykios, V. (2007). Duplicate record detection: A survey. *IEEE Transactions in Knowledge and Data Engineering*, 19(1):1–16.

Elmasri, R. and Navathe, S. (2000). *Foundamentals of Database Systems (3rd edition)*. Addison-Wesley.

European Commission (15 July 2008). Commission Regulation (EC) No. 665/2008 of 14 July 2008. *Official Journal of the European Union*.

Fan, W., Geerts, F., and Jia, X. (2008). Semandaq: A data quality system based on conditional functional dependencies. In *Proceedings of the Int'l Conference on Very Large Data Bases*, pages 1460–1463. VLDB Endowment.

Ginsburg, S. and Hull, R. (1983). Order dependency in the relational model. *Theoretical Computer Science*, 26:149–195.

Golab, L., Karloff, H., Korn, F., and D., S. (2010). Data Auditor: Exploring data quality and semantics using pattern tableaux. *Proceedings of the VLDB Endowment*, 3(2):1641–1644.

Hellerstein, J. (2008). *Quantitative data cleaning for large databases*. Report for the United Nations Economic Commission for Europe (UNECE), 42 pp.

Hipp, J., Güntzer, U., and Grimmer, U. (2001). Data quality mining - making a virtue of necessity. In *Proceedings of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 52–57.

Juran, J. (1964). *Managerial breakthrough*. McGraw-Hill, New York.

Lee, Y., Pipino, L., Funk, J., and Wang, R. (2006). *Journey to Data Quality*. The MIT Press.

Ng, W. (1999). Order functional dependencies in relational databases. *Information Systems*, 24(7):535–554.

Pivert, O. and Prade, H. (2010). Handling dirty databases: From user warning to data cleaning - towards an interactive approach. In Deshpande, A. and Hunter, A., editors, *Proceedings of the Int'l Conference on Scalable Uncertainty Management*, volume 6379 of *LNAI*, pages 292–305. Springer-Verlag.

Renear, A., Sacchi, S., and Wickett, K. (2010). Definitions of dataset in the scientific and technical literature. In Grove, A., editor, *Proceedings of the American Society for Information Science and Technology Annual Meeting*, volume 47(1), pages 1–4.

Shankaranarayan, G., Wang, R. Y., and Ziad, M. (2000). Modeling the manufacture of an information product with IP-MAP. In *Proceedings of the Int'l Conference on Information Quality*, pages 1–16.

Shanks, G. and Corbitt, B. (1999). Understanding data quality: Social and cultural aspects. In *Proceedings of the Australasian Conference on Information Systems*, pages 785–797.

Song, S. and Chen, L. (2011). Differential dependencies: Reasoning and discovery. *ACM Transactions on Database Systems*, 26(3), 16.

Szlichta, J., Godfrey, P., and Gryz, J. (2012). Fundamentals of order dependencies. *Proceedings of the VLDB Endowment*, 5(11):1120–1231.

van der Vlist, E. (2007). *Schematron*. O'Reilly Media.

Vassiliadis, P. (2009). A survey of Extract-Transform-Load technology. *Int'l Journal of Data Warehousing & Mining*, 5(3):1–27.

APPENDIX

IP-MAP

The IP-MAP graphical model is aimed at creating a systematic representation for capturing the details associated with the manufacturing of an information product. An information product is produced by means of processing activities and data quality checks on raw data and semi-processed information. Eight construct blocks are the main components of an IP-MAP: source block, customer block, data quality block, processing block, decision block, data storage block, organizational boundary block, and information system boundary block. Each construct block is identified by a unique name and is further described by a set of attributes (i.e., metadata). Figure 2 lists the symbols assigned to each construct block.

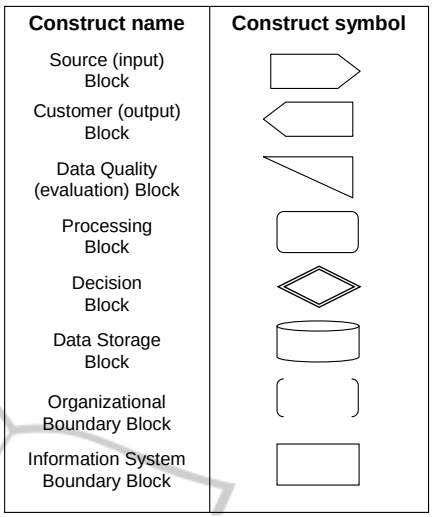

Figure 2: IP-MAP construct symbols.

Document Type Definition

```
<!DOCTYPE rules [
<!ELEMENT rules (rule_definition+)>
<!ELEMENT rule_definition (table_name+,
 (rule_cr|rule_fd|rule_od|
 rule_dd|rule_ec|rule_cc))>
<!ELEMENT rule_cr (if, then)>
<!ELEMENT rule_fd (lhs, rhs, when?)>
<!ELEMENT rule_od (lhs, rhs, when?,
 partition_on?)>
<!ELEMENT rule_dd (lhs_dd, rhs_dd, when?,
 partition_on?)>
<!ELEMENT rule_ec ((column+|(lhs_ec, rhs_ec)),
 when?, join_on?)>
<!ELEMENT rule_cc (check, join_on?)>
<!ELEMENT if (condition|conditions)>
<!ELEMENT then (condition|conditions)>
<!ELEMENT when (condition|conditions)>
<!ELEMENT check (condition|conditions)>
<!ELEMENT join_on (column+)>
<!ELEMENT lhs (column_name+)>
<!ELEMENT rhs (column_name+)>
<!ELEMENT lhs_ec (column)>
<!ELEMENT rhs_ec (column|(table_name?,
 disj_attr+))>
<!ELEMENT lhs_dd (distance_condition+)>
<!ELEMENT rhs_dd (distance_condition+)>
<!ELEMENT partition_on (column_name+)>
<!ELEMENT column (table_name?, column_name)>
<!ELEMENT disj_attr (column_name+)>
<!ELEMENT distance_condition (column_name,
 comparison_operator, constant)>
<!ELEMENT conditions ((condition|conditions)+,
 boolean_operator)>
<!ELEMENT condition (lside,
 comparison_operator, rside)>
<!ELEMENT lside (column|
 arithmetic_computation)>
<!ELEMENT rside (constant|column|
 arithmetic_computation)>
<!ELEMENT arithmetic_computation ((constant|
 column|arithmetic_computation)+,
 arithmetic_operator)>
<!ELEMENT table_name (#PCDATA)>
<!ELEMENT column_name (#PCDATA)>
<!ELEMENT comparison_operator (#PCDATA)>
<!ELEMENT boolean_operator (#PCDATA)>
<!ELEMENT arithmetic_operator (#PCDATA)>
<!ELEMENT constant (#PCDATA)>
<!ATTLIST rule_definition name CDATA #IMPLIED
 type (conditional_rule|functional_dependency|
 order_dependency|distance_dependency|
 existence_constraint|check_constraint)
 #REQUIRED>
<!ATTLIST rule_od type (direct|inverse)
 #REQUIRED>
<!ATTLIST rule_ec type (ec_dep|ec_bidir|
 ec_disj|ec_attr) #REQUIRED> ]>
```