ACTIVE REPAIR OF DATA QUALITY RULES

(Research-in-Progress)
(Methods, concepts, and tools for IQ)

Fei Chiang
Department of Computer Science
University of Toronto, Canada
fchiang@cs.toronto.edu

Renée J. Miller
Department of Computer Science
University of Toronto, Canada
miller@cs.toronto.edu

Abstract: The use of data quality rules, which capture business rules and domain constraints, is central to most data quality processes. Poor data quality often arises when the data and these rules (which are meant to preserve data integrity) become inconsistent. To resolve inconsistencies, organizations often implement specific, sometimes manual, sometimes computer-aided, cleansing routines to fix the errors. This solution necessitates frequent repetition of the data cleaning to resolve inconsistencies continually as the data evolves or grows. It is important to recognize that modern organizations may be as dynamic as their data. The business rules, application domain constraints, and data semantics will evolve. As business policies change, as data is integrated with new sources, and as the underlying data evolves, it becomes necessary to manage, evolve, and repair both the data and the rules.

In this work, we present a new data quality paradigm that uses automated support for both data and data quality rule repair and management. Previous techniques have focused mostly on updating or correcting the data. In contrast, our approach looks for clues in the data to understand if the data semantics or rules may have evolved. The approach is a holistic one that is designed to facilitate the continuous curation and maintenance of both data and data quality rules. A unique feature of our approach is that we use data mining to discover trends, contextual information, and data patterns that may yield meaningful insights into how a business rule has evolved. Our approach is designed to consider the very wide (many attribute) entity types or tables that are managed by many organizations. We recognize that due to acquisitions or business evolution, data quality rules may need to be evolved to account for many new features (attributes) of the data. We conduct two case studies using real business datasets that demonstrate the quality and usefulness of our methods in a continuous data quality process that manages and evolves both data and data quality rules. The evaluation provides promising results that show how a business analyst can use our tool to quickly identify errors, and identify if the errors are due to dirty data or to erroneous data quality rules that may need to be evolved. This understanding results in both improved overall data quality, and improved rule quality for better maintenance of new data.

Key Words: data quality, information quality, data inconsistencies, rule repair, data repair

INTRODUCTION

Poor data quality is a pervasive and serious problem for organizations, leading to inefficient operations, increased costs, and missed sales opportunities. It is estimated that over the next few years more than 25% of critical data in Fortune 1000 companies will continue to be inaccurate and incomplete [12]. Recent legislation such as the US Sarbanes-Oxley Act, the European Basel Accords, and the Privacy Act in Australia require organizations to maintain accurate and compliant data records. To manage this problem, companies have focused on correcting data anomalies by implementing specific, often manual, data cleansing routines [1, 6]. This solution however only solves part of the problem. First, data are not static and evolve over time due to various factors. A simple one time update solution will necessitate frequent updates each time there is a data inconsistency. The need for continuous data quality has lead to automation, that is, data mining techniques that can help find and, in some cases, suggest corrections for data inconsistencies. Secondly, implementing a data quality management process requires looking at the entire process from data capture down to the data itself. Personnel at all levels (from the upstream data capture, curation and design stages) down to the database administrators who manage the data directly, have to be involved to own and manage the information as a corporate asset. This involves elicitation and management of not only data, but data quality requirements.

Data quality requirements are used to define a set of data quality constraints (also known as integrity constraints) which are used to preserve data integrity. These constraints represent domain specific rules that should hold over the data and are defined according to the application semantics. Often these constraints are defined at design time by a data designer. Ideally, if the upstream data capture process is strict, and these constraints are enforced, then the intended data semantics are captured and preserved. In reality however, this is often not the case. Organizations may have weak enforcement of constraints for performance reasons or due to organizational or IT boundaries and limitations. If constraints are not enforced, data may become inconsistent. In addition, in modern applications, constraints evolve over time as applications and business rules change, as data is integrated with new sources, or as the underlying data semantics change. In these scenarios, the outdated constraints should be modified to reflect the intended data semantics and to support the business objectives. This correction process is normally performed manually by an analyst familiar with the domain and intended use of the data. This process can be very labor-intensive and prone to human error. If this is done manually, it may be hard for an analyst to gather all the information needed to make correct decisions.

Our goal is to develop a tool that can be integrated into an organization's data quality management process, and used to ensure that both the data and the data quality constraints remain correct and mutually consistent. We describe an application that highlights the benefit and usage of our repair tool next.

Application Area

Consider a European commercial banking institution, Credit Faire, wishing to expand their business to the US to increase their North American presence. Credit Faire decides that the easiest way to enter the US market is to acquire and merge with an established US bank named Upper North. Different countries have different financial regulations, banking policies, customer expectations, and product offerings.

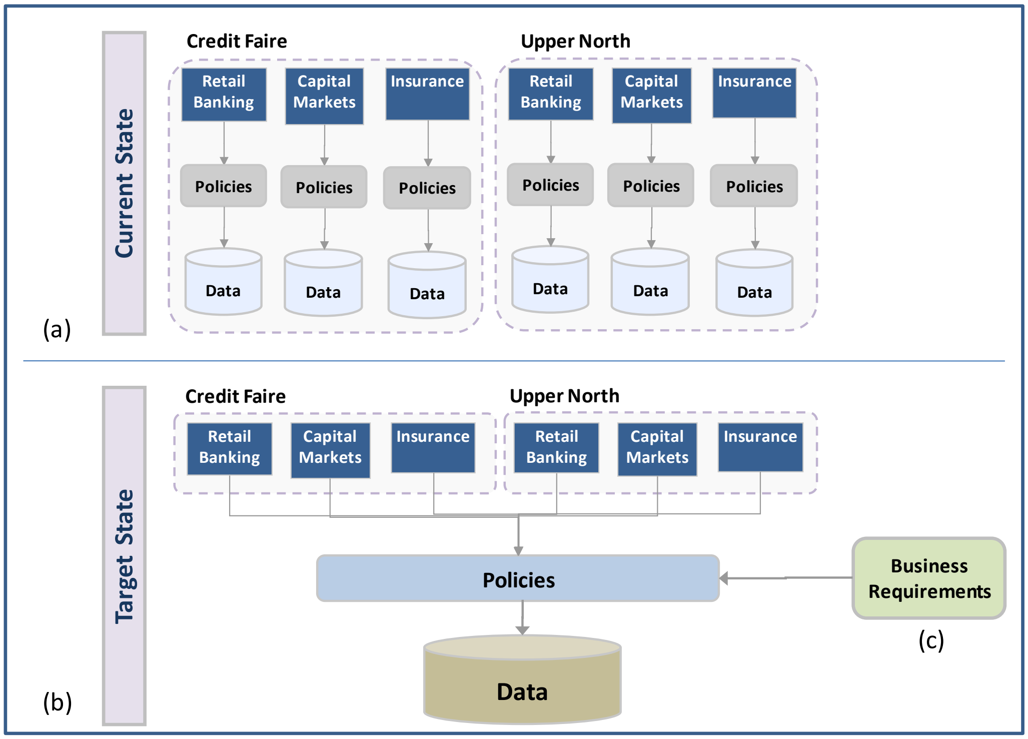


Figure 1: Example application

To complete this acquisition, the data models and systems of Upper North need to merge with those of Credit Faire, as shown in Figure 1(a). For simplicity, we list the major lines of business (LoB) of most major financial institutions (retail banking, capital markets, insurance), whereby each LoB has their own policies that act directly on their own data sources. The goal is to merge the data into one centralized data repository and one set of coherent policies as shown in Figure 1(b). In order to do this, an analyst would first need to review, understand, and integrate all the data instances into one central repository. To merge the policies, the analyst would then need to consult each LoB to understand their policies, and determine if there is overlap among policies, which ones are needed, and perhaps a prioritization of policies. Clearly, this can take a considerable amount of time, especially if the analyst has limited domain knowledge. Normally, consultants are engaged in this process, which can lead to increasing costs. Our constraint repair tool can help such an analyst and consultant by reducing the investigation and consulting time, by analyzing the set of constraints (policies) and determining their relative strength against the data sources. Once the problematic/outdated constraints are identified, the tool proposes a set of repairs to these policies (such as adding new fields or data values that reflect the current semantics). An analyst can then review these solutions to determine the ones that are most appropriate for the application. By shifting the integration task to the repair tool, which searches for overlapping policies across LoBs, analysts are able to offer more value by verifying the recommended solutions. The repair tool focuses the analysts attention on constraints or constraint repairs that are supported by the data. In addition, having a streamlined set of policies across LoBs improves operating efficiency and reduces costs.

Our repair tool can be integrated into a continuous data quality management process and can be used routinely to detect outdated policies resulting from changes to the business requirements (Figure 1(c)). For example, the recent economic crisis in the US introduced strict conditions on lending requirements for consumers. These new borrowing conditions are not always applied pervasively throughout an organization, especially for large, silo environments. In these cases, our repair tool can assist a broker who is handling a lending file to identify the current policies that are inconsistent, and recommend repairs. This saves considerable investigation time for the broker (to discover the problematic rules), and reduces the processing time by ensuring consistent policies across different lines of business.

Previous work in this area has focused on data repairs, by finding low cost changes to the data [2, 5, 8, 11]. These techniques assume the given set of constraints is correct (and hence are not updated), making the techniques inflexible to changes in business policies. Chiang and Miller [4] present a unified repair model that considers data repairs and constraint repair equally to resolve an inconsistency. Their constraint repair model is limited to adding only a single field (attribute) to the constraint, which can be insufficient to correctly express the intended semantics. Furthermore, the application of their work was towards traditional databases and functional dependencies. In contrast, our focus is on practical business applications using (more general) data quality rules. We present new techniques for finding context, that is the scope under which a data quality rule holds.

We make the following contributions:

- An extended repair model that analyzes and recommends a set of attributes to add to a data quality rule such that it is consistent with respect to the given data instance.
- A discovery tool that searches and identifies the conditions (data values) where a data quality rule holds over a subset of the table. Our tool is able to identify data patterns that are most common with respect to the given rule.
- An extensive experimental evaluation, focusing on practical business cases, that highlights the usefulness of our repair tool, and the quality of the generated recommendations.

A data quality solution has many properties including the following [12].

- Existence: whether the organization has the data.
- Validity: whether the data values are within the acceptable domain.
- Consistency: whether the same piece of data contains the same value across different locations.
- Integrity: the completeness of relationships between data elements.
- Accuracy: whether the data describes the properties of the object it is meant to model.
- Relevance: whether the data is appropriate to support the business objectives.

Addressing the existence and relevance properties is the first step towards implementing a data quality solution. The remaining four properties are often organization and department specific depending on the data usage. Our repair tool addresses the remaining four data quality properties in a cross-cutting way. The tool finds (and exploits) commonalities in data quality rules across the organization and finds scoping rules (conditions under which a rule applies) to permit specialization of data quality rules within a department or LOB.

- Validity: having a repaired set of correct constraints will ensure valid data values are captured.
- Consistency: if the constraints are correct and enforced, then data referring to the same entity should have the same value.
- Integrity: ensuring the data quality rules remain valid and complete is central to preserving data integrity.
- Accuracy: the repair tool will help an analyst to repair incorrect data values. The data repair tool uses evidence in the data to suggest likely correct repair values thus reducing an analyst’s time in searching for an accurate value.

Outline of this Paper

We first present preliminary background and notation regarding the types of constraints covered in this work and the repair operations. Next, we present the unified repair model. The repair operations we consider are derived from the needs of applications such as the one described above. In the evaluation section, we evaluate the usefulness of the model by using a domain expert. We conduct two case studies using real datasets with business applications. Finally, we summarize and conclude our work with some insights.

PRELIMINARIES

We begin by defining the types of constraints, constraint violations, and repairs that we will consider. In a database, we define an instance I of a relation over a set of attributes (also known as fields), where $N = |I|$ denotes the number of tuples in the relation, and n denotes the number of attributes. We use X and Y to denote attribute sets.

A data quality rule (or rule) $F: X \rightarrow Y$ is specified over two sets of attributes X and Y. For example, *F:* [JOB_ROLE, LOCATION] → [SALARY_RANGE] states that for a given job at a given location there is at most one acceptable salary range. A data instance I satisfies F, denoted $I \models F$, if for every pair of tuples $t1, t2$ in I, if $t1[X] = t2[X]$ then $t1[Y] = t2[Y]$. An instance I is *inconsistent* with respect to (wrt) F if it violates F. For example, if I contains the two tuples *t1*: (‘manager’, ‘new york’, 80k-100k) and *t2*: (‘manager’, ‘new york’, 120k-130k), then I would be inconsistent wrt F, since the manager role in new york does not have a unique salary range. We assume (without loss of generality) that for all rules $F: X \rightarrow Y$, Y contains a single attribute. In addition, rules may be conditional (or context dependent). Hence, any attribute mentioned in a rule may also have a condition specified on it of the form [<attribute> <predicate> <value>]. Such conditional rules can be used to model domain constraints, for example [SALARY] < [100k]. A domain constraint is violated if for any tuple, the attribute’s data value lies out of the validated range. More general conditional rules include a rule that states that within Canada, a postal code determines a unique street *[Country = ‘Canada’, PostalCode]* → *[Street]*. This rule states that the (non-conditional rule) [PostalCode] → [Street] holds, but only for tuples with Country value of ‘Canada’. Note that a rule without conditions is simply a *functional dependency* (well-known in the relational database literature), while a rule with one or more conditions is a *conditional functional dependency* [3] which have shown to be very useful for data cleaning and for improving data quality [7] by allowing a data analyst to express greater semantics according to their application domain.

Data Repairs

Following the data repair model of Chiang and Miller [4], suppose we have tuples *t1* and *t2* that violate $F: X \rightarrow Y$, where $t1[X] = t2[X]$, but $t1[Y] \neq t2[Y]$. To repair this violation, we can update the attribute values in the following ways (assuming *t2* is updated, without loss of generality):

1) Update the Y values to be equal: we would update $t2[Y]$ to be equal to $t1[Y]$, provided that there is sufficient evidence in I to support that $t1[Y]$ is the correct value.
2) Update the X values to be different: We would update $t2[X]$ to be a different value than $t1[X]$. We do this by selecting a value $t3[X]$ that is close to $t2[X]$ (defined according to a similarity measure such as edit-distance), and $t3[Y] = t2[Y]$, and there is sufficient evidence supporting $t3[X]$.

Rule Repairs

To repair violated rules, we consider (as in the unified repair model [4]) the addition of a single attribute to *X*. This extra attribute provides context to define when the rule actually holds. This is often needed when integrating disparate sources and an extra attribute is needed to distinguish the individual sources. Clearly, adding only a single attribute is not sufficient in many cases. We extend the repair operations to include multi-attribute repairs. This involves searching the space of attributes not in *F* and selecting a subset of these attributes to add to *X*. This operation provides increased flexibility to select the attributes that can correctly express the semantics needed to repair the rule. As data quality rules are often contextual, we also provide a new repair operation that adds context (a condition) to the rule.

REPAIRING INCONSISTENT RULES

We apply the repair model of Chiang and Miller [4] to derive the data repairs and the single attribute repairs. First, we will briefly describe this repair model, followed by our extensions that introduce multi-attribute repair and adding context to data quality rules.

A Unified Repair Model

The unified repair model [4] is based on the Minimum Description Length (MDL) Principle which states that the best model for a dataset is the one that can take advantage of the regularity in the data (leading to the best compression), and resulting in the smallest model. In other words, it's the simplest model that can encode the data the best. The description length is a quantification of this idea defined as the sum of the length of the model and the length of the data given the model. Chiang and Miller exploit the regularity in the data and use the MDL principle to quantify the consistency in the data by using the description length as the basis for their cost function. They search for a model that can represent as much of the data as possible. For inconsistent tuples with respect to a rule *F*, they evaluate a set of data repairs and rule repairs. Data repairs that exploit and maximize the regularity in the data, and are of low update cost (i.e., the source and target data values are similar), are recommended to the user. For rule repairs, all attributes that do not belong to $F: X \rightarrow Y$ are evaluated as candidates to add to *X*. The single attribute whose values neatly separate the conflicting tuples and preserves the existing data regularity is chosen as the winner. For example, the constraint *F*: [JOB_ROLE] → [SALARY_RANGE] may cause 30% of the tuples to be inconsistent in *I*, and contain 70% satisfying tuples consisting of, say, 50 distinct pattern values (a pattern value is a set of data values such as ('secretary', 30k-50k)). If we consider adding the employee's years of service as a rule repair, to get *F1:* [JOB_ROLE, YEARS-SERVICE] → [SALARY_RANGE], this may reduce the inconsistent tuples to only 15% (now 85% of the table is satisfied), but we may have 80 distinct pattern values since many of the original satisfying pattern values are now separated by the new attribute. In other words, the attribute YEARS-SERVICE reduced the regularity in the data wrt *F*. Another attribute such as SUBSIDARY or LOCATION may be a better choice to reduce the number of inconsistent tuples, and also preserve the regularity in the data. The rule repair search process aims to find such attributes.

Adding Multi-Attribute Repair

Adding a single attribute is a limited repair operation as some rules may require greater expressive power by adding multiple attributes. We extend the search to look for candidates involving a set of attributes that reduce the number of inconsistent tuples, while preserving as much of the regularity in the data as possible. To find these attributes, we apply ideas from reconstructability analysis, which we explain next.

Reconstructability Analysis

Given a relation and a set of attributes, known independencies among the attributes can be used to simplify the data and remove redundancy, giving a clear view of the data and the attribute relationships. Database normalization follows this principle. In reconstructability analysis [13], the independencies are not known *a priori*, and a search process attempts to infer simpler models by considering the trade-off between weak associations in the data against increasing complexity in the model. To find the best model, the analysis begins by simplifying the relation into a set of sub-relations containing a smaller number of attributes, which reduces the complexity. The set of sub-relations (attributes and data) is called the *model* of the overall relation, and also known as the *reconstruction hypothesis*. Since there are many hypotheses available, we can measure the information content in the reconstructed model and compare it against the information content of the original relation. If this distance is close, then less information is lost by using the reconstructed model. Selecting a suitable model is a trade-off between complexity and accuracy. The original relation is most accurate but also most complex, versus less accurate models that are simpler but enable us to better understand the structure and relationships among the attributes. In summary, reconstructability analysis is the problem of decomposing an overall system into simpler sub-systems such that sufficient information is preserved (about the whole system) to facilitate deeper understanding of the system.

Multi-Attribute Repair Solution

Given a data instance *I* and a rule *F: X* → *Y*, we use OCCAM [14], a discrete multi-variate modeling tool developed by the Systems Science Department at Portland State University. OCCAM is based on reconstructability analysis, and we use it to determine the set of attributes to add to *X*. We specify the set of attributes in *F* as a starting model, noting the independent variables *X* and the dependent variable *Y*, and OCCAM will perform an upwards search from this starting model to find the simplest model with high information content. The difference in information content between the reconstructed model and the original model is evaluated as a difference of distributions. This difference must be statistically significant in order to validate the quality of the reconstructed model. We take this reconstructed model and extract the attributes not in *F* as the candidate set of attributes to consider in the description length cost function. If the description length cost of adding these attributes is lowest among all other types of repairs, then this multi-attribute solution is the recommendation to the user to resolve the inconsistency. We report quality results in the evaluation section.

Adding Context to Data Quality Rules

Data quality rules are often contextual where there are specific values that the rule holds. For example, a policy at a bank may state that if a customer has been doing business with the bank for over 10 years, and they have assets over $300k, the bank offers the largest discount on current mortgage rates. The loyalty and asset value conditions may not always be explicitly specified, since much of the time these decisions are at the discretion of the individual or department. However, general rules normally exist to indicate that such a relationship among attributes exists, such as *F: [LOYALTY, ASSET_VALUE]* → *[DISCOUNT_RATE]*. As such rules evolve over time, they may only be satisfied under particular conditions. For example, for loyal customers of over 10 years a discount rate is offered that is uniquely determined by the customer's asset value *C1: [LOYALTY > 10, ASSET_VALUE]* → *[DISCOUNT_RATE]*. Alternatively, the evolved dataset may also contain useful information on common data patterns. For example, we may observe that for loyal customers, many banking managers have given customers with assets less than $100k the maximum discount rate, which may not be desirable. Alternatively, we may find that a more specific rule such as *C2: [LOYALTY > 10, ASSET_VALUE < $100k]* → *[DISCOUNT_RATE = maximal]* (which is an example of a constant rule) now holds over the data, while the more general rule C1 is no longer an accurate description of the data.

Given an inconsistent data instance and a set of rules, our repair tool searches for common data patterns, and for conditions where these rules may hold over the data. These contextual rules can provide meaningful information about the specific conditions under which they hold. In addition, the data patterns can reveal common trends and identify entities and their properties that were previously unknown. We present these data patterns and contextual rules to a domain expert (user) who can validate their correctness and accuracy. The user can update or apply these findings at their discretion, since data quality and application requirements are often subjective requiring interactive user feedback.

To find the context (conditions) under which a rule *F* holds more firmly, we identify all the satisfying tuples in *I* wrt *F*. For each set of candidate attributes, and each value in its domain, we cluster all the tuples that contain the same data value, resulting in *k* clusters for a domain size of *k*. We check if any of these clusters contain *all* the satisfying tuples. If there is more than one such cluster, we select the cluster of largest size *k*-max. The data value *v* represented by cluster *k*-max is the condition where *F* partially holds over *I*, resulting in a rule of the form [A = 'v', X] → Y, where A is the set of candidate attributes.

If no conditions can be found satisfying the minimum support threshold (i.e., none of the clusters contain all the satisfying tuples), our repair tool will return the constant rule with the largest support. This data pattern can provide insights about the represented entities and their properties. We note that contextual rule repairs are not necessarily a subset of multi-attribute repairs. We may find a contextual rule that contains constant values with respect to the original attributes, that is, there are no additional attributes added.

Given a minimum support threshold (that indicates the minimal amount of evidence in the data to support the rule), we partition the tuples in *I* according to the values of *XY* (the attributes of *F*). If *XY* has a domain size of *m*, we have *m* partitions. We select the largest partition and check if it is greater than the support threshold. If so, the data values represented by this partition, say '*a*' and '*b*', are returned as the constant rule of the form of *[X = a]* → *[Y = b]*. The data patterns found in a constant rule can be used to detect inconsistencies wrt domain constraints, by validating whether the data values conform to those in the domain constraints. This automatic detection and notification allows a more experienced data analyst to take the necessary steps to correct the inconsistency either by modifying the domain constraint values or the actual data values. We present examples of discovered rule repairs in the evaluation section.

EVALUATION

We conduct a qualitative evaluation of our repair operations. We employ the existing implementation of the unified repair model in C [4], and add our repair extensions (also written in C). Our experiments were run on a Dell PowerEdge R310 with 64GB of memory, quad core X3470 2.93 GHz processor, running Ubuntu 10.04. We first describe the datasets used in our experiments, and then report qualitative results from our study.

Data Characteristics

We use two real world datasets. We conduct two business case studies in the areas of:

1. Vehicle fuel efficiency (vehicles dataset), and
2. Regulation of financial institutions (finance dataset)

The vehicles dataset is based on the US Environmental Protection Agency's Green Vehicle Guide that provides vehicle ratings based on emissions and fuel economy [10]. The dataset we use contains 3000 tuples and 17 attributes, as described in Table 1. In the second case study, we investigate the factors influencing the operation and regulation of financial institutions. We use the US Federal Deposit Insurance Corporation (FDIC) directory [9] to get a list of all FDIC insured institutions. The data contains information about the institution such as type, asset value, regulators, geographic information (city, state), and operating status (active, bank class). The dataset contains 30,000 tuples, with a schema given in Table 2.

The goal of the case studies is to evaluate the usefulness and quality of the generated repairs. We compute the precision and recall of the data repairs, the single attribute, and the multi-attribute repairs. We employ a domain expert to review the list of recommended repairs to mark the correct repairs, and to identify the most interesting rules and data patterns found.

Case Studies

To assess the correctness and usefulness of the recommended repairs, we conduct two case studies that evaluate the repair quality. We compute the precision and recall of the data repairs, and the rule repairs as defined below.

Attribute	**Description**
Model	Model of the vehicle
Displ	Engine size
Cyl	Number of cylinders
Trans	Transmission type
Drive	Type of drive
Fuel	Type of fuel
Sales-Area	State area
Stnd	Code for the emission rating
Stnd-Desc	Description of the emission rating
Underhood-ID	Vehicle identification
Veh-Class	Class of the vehicle
Air-Poll	Air pollution score (10 = best)
City-MPG	Mileage per gallon in city driving
Hwy-MPG	Mileage per gallon in highway driving
Cmb-MPG	Fuel economy
Greenhouse	Greenhouse gas score (10 = best)
Smartway	Is the vehicle a top environmental performer

Table 1: Schema of the vehicles dataset

Attribute	**Description**
Stname	State name
FHFB_member	Member of the federal housing finance board
Active	Institution is open and insured by FDIC
Address	Street address of head office or a branch
Assetvalue	Asset value categorization
Bkclass	Type of bank
Changec1	Code of a structural event relating to the institution
Conserve	Operated in government conservatorship
Adj_deposit	Total deposits
Adj_equity	Equity capital
Inactive	Institutions closed but previously insured
Instcrcd	Credit card institutions
Insfdic	Insured by FDIC
Inssaif	Members of savings association insurance fund
Name	Legal name of the institution
Adj_roa	Categorization of the return on assets
Stlp	State alpha code
Zip	Zipcode
City	City where headquarters are located
Regagnt	Federal regulator
Otsregnm	Office of the supervision region
Adj_depdom	Domestic deposits
Adj_netinc	Adjusted net income
Stmult	Does the institution operate interstate branches
Specgrpn	Primary specialization

Table 2: Schema of the finance dataset

$$\text{Precision(data)} = \frac{\#\text{correct repairs}}{\#\text{returned repairs}} \qquad \text{Recall(data)} = \frac{\#\text{correct repairs}}{\#\text{errors}}$$

$$\text{Precision(rule)} = \frac{\#\text{correct tuples}}{\#\text{modified tuples}} \qquad \text{Recall(rule)} = \frac{\#\text{correct tuples}}{\#\text{error tuples}}$$

A domain expert reviews the recommended data repairs and marks those that are correct. To determine the total number of errors, we take a random sample of the dataset, and the domain expert identifies the total number of existing data errors from this sample. For rule repairs, the #correct tuples are tuples that are resolved wrt *F* by adding the recommended attributes. If the recommended attributes *A* have a large domain (contains many distinct values), then adding the set *A* to *F* may not neatly separate the conflicting tuples, but may also separate consistent tuples. The #modified tuples are the number of tuples that have been separated by *A*, regardless of whether they were inconsistent.

Precision and recall values range from 0 to 100%. Precision measures the percentage of returned repairs that are correct, hence 100% precision is best. Recall measures the completeness of the returned repairs, i.e., what percentage of the total errors did the recommended repairs capture? Again, 100% recall is best indicating that all the errors were accounted for in the results. We do not compute precision and recall for contextual and constant rules since these rules only hold for a subset of the relation, and all the tuples in a contextual/constant rule are consistent.

Case 1: Vehicle Fuel Efficiency

In the first case study, we use the vehicles dataset based on the US Environmental Protection Agency's Green Vehicle Guide that provides vehicle ratings based on emissions and fuel economy. We postulate the set of rules shown in Table 3.

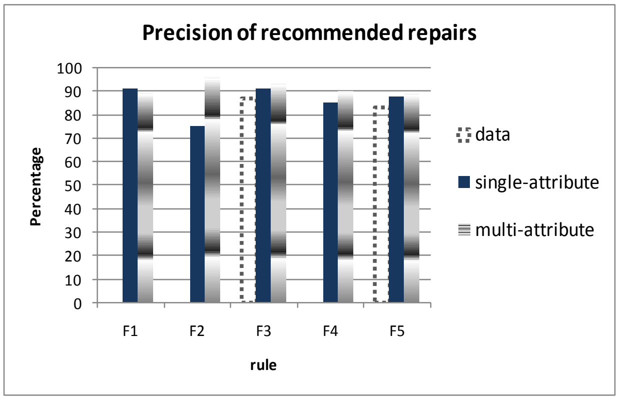
Rule	**Description**
F1: Sales-Area → Smartway	Do particular states promote more environmental friendly cars?
F2: Model → [Cyl, Trans]	The car model determines the transmission and number of cylinders
F3: [Fuel, Cyl, Drive] → Cmb-MPG	Cylinders, fuel, and drive type determine mileage
F4: [Cmb-MPG, Cyl, Drive] → Veh-Class	Mileage, cylinders and drive specs are factors to determine the class of car (e.g., SUV, compact car)
F5: Air-Poll → Greenhouse	Air pollution score determines the greenhouse score

Table 3: Rules on the vehicles dataset

We run our tool using these rules to detect and repair any inconsistencies between the rules and the data. We envision that a business analyst may also follow this routine as part of their data quality management process to detect any inconsistencies. The analyst may run the repair tool on a snapshot of the data (for example every quarter), after the business has assessed their financial position and re-adjusted their objectives and policies. The analyst would need to load the raw data file, and select the appropriate parameter values. In our case, the repairs were generated in 8.1 secs, and returned a list of data and rule fixes to resolve the inconsistencies and errors.

The repair tool returns a list of repairs categorized by type, i.e., data, single attribute, multi-attribute, and contextual. We do not compute precision for contextual repairs since all the tuples in these rules are consistent. Figure 2 shows the precision values for the data, single attribute, and multi-attribute repairs. There were no data repairs generated for F1, F2, and F4. Overall, the multi-attribute repairs showed the best (consistent) precision across all five rules. This is due to the greater flexibility and expressiveness that adding a set of attributes can provide versus adding only a single attribute. In particular, multi-attribute repairs were the preferred repair option for F1, F2, and F4. This indicates that the recommended attributes were semantically correct to add to F, and they resolved the largest number of inconsistencies by separating the inconsistent data values. For F3 and F5, data repair and single attribute repair were the recommended solutions, respectively. Overall, the precision values show that the repair tool and the extended multi-attribute repair propose meaningful repairs with precision values primarily 80% or higher.

After validating the repair correctness with precision, we wanted to determine how many errors the generated repairs captured. To do this, we compute the recall, as shown in Figure 3. The recall values do not show a dominant type of repair. Data repair showed the best recall for F3 and F5 due to the increased number of domain values of Cmb-MPG, Air-Poll, and Greenhouse. F1 and F2 showed the best recall with single attribute repair, while F4 did the best with multi-attribute repair. Overall, the results show that approximately 77% of the total errors are captured by all the recommended repairs.



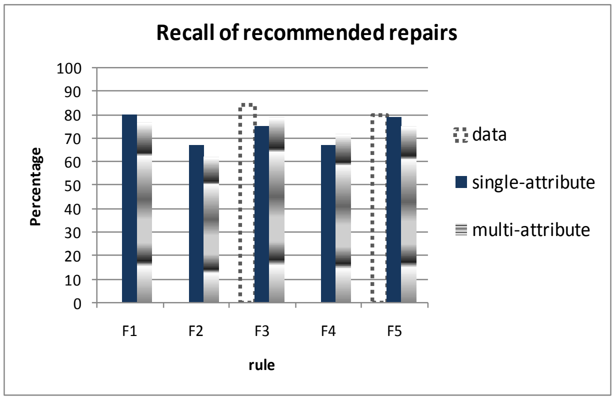


Figure 2: Precision values for the vehicles dataset

Figure 3: Recall values for the vehicles dataset

Sample Repairs

Since inconsistencies were found between the data and the rules, we take a closer look at the recommended repairs for each rule. F1 determines which vehicle attributes influence environmentally friendly vehicles. Different states and sales areas will have different regulations and policies supporting eco-friendly vehicles. Our repair tool recommended adding the attribute Stnd (the vehicle emissions level rating) to F1 to distinguish its eco-friendly status. This is clearly an intuitive choice. The contextual rule repair found an interesting data pattern that indicates in California, most vehicles must satisfy the California LEV-II program requirements. This information can be used as a formal rule to identify non-conforming vehicles and flag them for review. In F2, the winning multi-attribute repair was to add Cmb-MPG and Drive to F2, to better distinguish the cylinder and transmission values. These attributes preserved the regularity in the data by splitting the fewest number of consistent tuples. The contextual rule repair found that the Ford F350 with 8-cylinders and automatic transmission was the most common vehicle satisfying this rule.

Data repairs are recommended to resolve the inconsistencies in F3, and to impute the missing data values. In particular, inconsistencies exist for the following models and attributes: for the Dodge Ram 3500 and Volkswagen Jetta (Sportswagen), the fuel type needs to be updated from gas to diesel, the Honda Element’s drive type should be updated to 2WD, and the Ford F250, F350 both need the Cmb-MPG values populated. However, data repairs are not sufficient to resolve all the inconsistencies, and the Model attribute should also be added to F3. This repair is intuitive since the mileage varies based on the car model. Applying this repair resolves 1350 tuple inconsistencies. For F4, adding the vehicle Model and transmission (Trans) was the recommended repair. The contextual rule repair identified small cars with 2WD and 22 mpg as the most popular in this data. Lastly, F5 states that the air pollution score directly influences the greenhouse score. There were 2650 data values that required updating, primarily focused on the greenhouse scores. The data repairs revealed that many luxury cars (e.g., Aston Martin V8 Vantage, Audi R8, BMW M3) have poor greenhouse ratings equal to 3. Furthermore, the vehicles Ferrari, Bentley Continental, Rolls Royce Phantom, and Lamborghini Gallardo all have models that possess the worst rating equal to 1. This information can be useful to these vehicle manufacturers to greatly improve their environmental platform. The multi-attribute repair found that adding Drive and Cmb-MPG would help distinguish the greenhouse score. Finally, our contextual rule repair found that under the condition Cmb-MPG = 18, this rule held with about 200 supporting tuples, indicating that 18 mpg is an ideal mileage level among vehicle manufacturers. Table 4 summarizes the recommended repairs, and gives a sample of the discovered data patterns, and the contextual rule for F5.

Rule	**Type of Repair**	**Solution**	**Contextual rule data pattern**
F1	Single attribute	Stnd attribute	In California, cars must satisfy California LEV-II emission level requirements.
F2	Multi attribute	Cmb-MPG, Drive attributes	Ford F350 truck with 8-cyl, auto transmission was most popular satisfying vehicle.
F3	Data, single attribute	Dodge Ram 3500 gas → diesel, Model attribute	[gas, 6cyl, 2WD] → [21mpg] is the most popular spec
F4	Multi attribute	Model, trans attributes	[22mpg, 6cyl, 2wd] → [small-car] is most popular
F5	Multi attribute	Drive, Cmb-MPG attributes	[Cmb-mpg = 18, Drive, Air-poll] → [Greenhouse] is a well supported rule at this mileage level.

Table 4: Summary of repairs for vehicles dataset

Case 2: Regulation of Financial Institutions

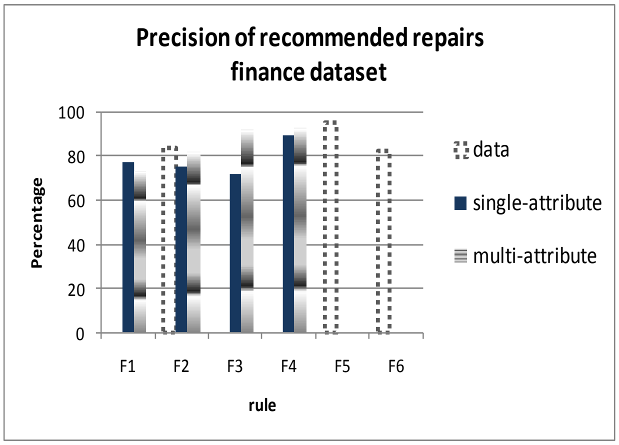
In the second case study, we use a financial dataset from the US Federal Deposit Insurance Corporation (FDIC) directory [9] that lists all FDIC insured institutions. The data contains information about an institution such as type, asset value, regulators, geographic information, and operating status. We postulate the set of rules shown in Table 5.

Rule	**Description**
F1: City → Otsregnm	The headquarters city determines the office of the supervising region
F2: Adj_equity → Adj_deposit	Equity capital determines total deposit amount
F3: [Stname, Specgrpn, Bkclass] → Stmult	The state location, bank specialization, and class of bank determines whether it operates inter-state banking services
F4: Conserve → Adj_equity	If the institution is in government conservatorship, it can influence its equity holdings
F5: Bkclass → Regagnt	The type of bank determines the regulating body
F6: Changec1 → Assetvalue, Adj_roa	Re-structuring activity influences the bank's asset value and return on assets

Table 5: Rules on the finance dataset

Our repair tool took approximately 6 mins to generate a set of repairs. Figure 4 shows the precision values for the data, single attribute and multi-attribute repairs. There were no data repairs generated for F1, F3, and F4. For F5 and F6, there were very few tuples corrected by attribute repairs, hence there are no precision results for this repair type.

Figure 4 shows that when the multi-attribute repair was available, it returned results that were almost equal or better than the single attribute repair. Data repairs were favoured against multi-attribute repairs for rules having large domains. This is because it was cheaper to simply update the anomalies and preserve the regularity in the data versus changing the rule. Overall, data repairs were recommended for F2, F5, and F6. Multi and single attribute repair was recommended for F3, F4, and F1, respectively. The precision values show that at least 75% of the repairs are correct, achieving as high as 95% correctness in F5. In particular, multi-attribute precision values are consistently 77% or higher, showing their usefulness to generate correct repairs.



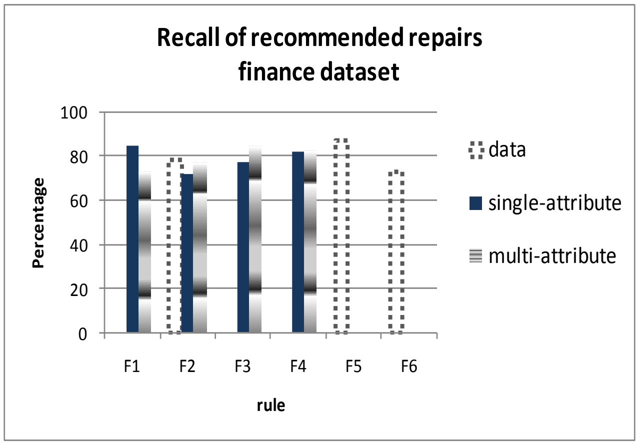


Figure 4: Precision values for the finance dataset

Figure 5: Recall values for the finance dataset

Figure 5 shows the recall values across all six rules. None of the repair types is a consistent winner among all the rules. For rules where a data repair was suggested, the data repair was also the best solution. This occurred for F2, F5, and F6, where specific data updates capture more errors inherent in the large domain size of the participating attributes. The multi-attribute repair option was best for F3 and F4, where the recommended attributes helped to separate the problematic values, and fix over 83% of the erroneous tuples.

Sample Repairs

We take a closer look at the recommended repairs for each rule. F1 states that the bank headquarters city determines the supervising region. This rule is fairly clean except for some cities that span multiple regions. There were no data repairs reported, and the solution was to add the federal regulator (Regagnt). This is a correct choice since Regagnt and the supervising region (Ogsregnm) are correlated. The contextual rule results show that many financial headquarters are located in the Chicago region in the central US, and are regulated by the Office of Comptroller of Currency (OCC). In F2, the equity holdings of a bank determine their deposit amounts. This rule was generally true with some exceptions. The data updates corrected institutions that did not have as much equity and deposits as expected (i.e., decreasing the deposit and equity to lower amounts). The attribute repairs suggested Assetvalue and Adj_depdom, which were not as favorable as the data repairs since the rule was not largely inconsistent with the data.

Factors influencing whether a bank operates inter-state branches are explored in F3. Inconsistencies were resolved via multi-attribute repair by adding Inssaif (member of savings association) and City. Only stable institutions in particular cities have the capacity and demand to operate across state boundaries. Our contextual rule repair found that large, commercial lending institutions operate across different states since they possess the resources to do so. Multi-attribute repair was also suggested for F4 to add City and Regagnt (the regulating agency), to determine the equity holdings of a bank in conservatorship (under the control of a regulator/conservator). The contextual rule repair found that small equity banks were at most risk for conservatorship.

Finally, data repairs were the recommended solution for F5 and F6, since the attribute repairs corrected a minimal number of tuples. In F5, the suggested fix was to update 'savings bank' to 'commercial bank' for a number of institutions. The contextual rule data pattern shows that most institutions are commercial banks regulated by the FDIC. The attribute repair for F6 proposed adding State to distinguish a bank's asset and return on asset (roa) value, which only corrected 262 records. The data repair corrected more errors by updating each value on a case by case basis. For example, the asset value and roa were lowered for particular banks, and one institution had an incorrect re-structuring activity that indicated its products had merged, when in fact, the headquarters location had moved. Table 6 summarizes the recommended repairs, and gives a sample of the discovered data patterns.

Rule	**Type of Repair**	**Solution**	**Insights from contextual rule data patterns**
F1	Single attribute	Regagnt attribute	Many bank headquarters located in Chicago in the Central US, regulated by OCC.
F2	Data	Update the deposit and equity values to lower amounts	N/A
F3	Multi attribute	Inssaif, City attributes	Large, commercial lending institutions provide inter-state services
F4	Multi attribute	City and Regagnt attributes	Small equity banks were at most risk for conservatorship
F5	Data	'savings bank' → 'commercial bank'	Most institutions are commercial banks regulated by FDIC
F6	Data	Lower the Assetvalue, Adj_roa amounts, correct the restructuring activity	N/A

Table 6: Summary of repairs for finance dataset

From the two case studies, it is evident that the repair tool and our extensions provide significant value to a business analyst. First, our tool saves an analyst from the manual and tedious effort of updating the hundreds or thousands of outdated rules and incorrect data values that can exist after policy changes and as the data evolves. Second, meaningful insights are gleaned during the repair process as our contextual rule repair operation provides useful information on common data patterns that exist in the data, as well as the context under which certain rules may hold. This information can be cross-checked against business requirements to validate whether the data patterns are aligned with current business policies and objectives. Third, many of the suggested repairs involve multiple attributes, revealing that greater expressiveness is needed to correct outdated rules from the original repair model [4].

CONCLUSIONS AND INSIGHTS

In this paper we have discussed the problem of managing data inconsistencies with respect to a set of rules. Previous techniques have focused on one time solutions of correcting the data errors at hand.

However, modern applications require that both the rules and the data be maintained as business policies change, as data sources are integrated, and as data naturally evolves. We have presented an extended repair model that focuses on ensuring the set of data quality rules are correct. Our evaluation has shown that previous work of only adding a single attribute to a rule is insufficient to handle all application domains and types of rules. Our multi-attribute repair has shown to provide correct and complete repairs by providing greater expressiveness to correct outdated rules. In addition, our contextual rule discovery and repair yielded meaningful insights that businesses can use to refine their policies and goals. We envision that our repair tool can be integrated into an organization's data quality process that allows a data analyst to efficiently detect and resolve inconsistencies, as well as discover meaningful insights from the data.

ACKNOWLEDGMENTS

This work was partially supported by the NSERC Business Intelligence Network.

REFERENCES

[1] Batini, C. and Scannapieco, M. *Data Quality: Concepts, Methods and Techniques.* Springer Verlag, 2006.

[2] Bohannon, P., Fan, W., Flaster, M., and Rastogi, R. *A cost-based model and effective heuristic for repairing constraints by value modification.* SIGMOD '05.

[3] Bohannon, P., Fan, W., Geerts, F., Jia, X., Kementsietsidis, A. *Conditional functional dependencies for data cleaning*. ICDE 2007.

[4] Chiang, F., Miller, R.J. *A unified model for data and constraint repair.* ICDE 2011: 446-457.

[5] Cong, G., Fan, W., Geerts, F., Jia, X., Ma. S. *Improving data quality: consistency and accuracy.* VLDB '07.

[6] Dasu, T., Johnson, T. *Exploratory Data Mining and Data Cleaning.* John Wiley 2003.

[7] Fan, W. *Dependencies Revisited for Improving Data Quality.* PODS 2008 Tutorial.

[8] Fan, W., Li, J., Ma, S., Tang, N., Yu, W. *Interaction between record matching and data repairing.* SIGMOD 2011.

[9] FDIC Institution Directory: http://explore.data.gov/catalog/raw/?category=Banking%2C+Finance%2C+and+Insurance /

[10] Green Vehicle Guide: http://explore.data.gov/catalog/raw/?category=Transportation

[11] Kolahi, S., and Lakshmanan, L.V. *On approximating optimum repairs for functional dependency violations.* ICDT '09.

[12] Moore, M. *Dirty Data is a Business Problem, Not an IT Problem.* Gartner Research Report. 2007.

[13] Reconstructability Analysis Tutorial http://www-lehre.informatik.uni-osnabrueck.de/~ftprang/papers/tproject/node34.html

[14] Willett, K., Zwick, M. *A Software Architecture For Reconstructability Analysis*. Kybernetes, vol. 33, No. 5/6, pp. 997-1008. 2004. See also: OCCAM: A discrete multi-variate modeling tool based on Reconstructability Analysis - http://dmm.sysc.pdx.edu/