


<b>ITC 2/50</b> <b>Information Technology and Control</b> <b>Vol. 50 / No. 2 / 2021</b> <b>pp. 247-263</b> <b>DOI 10.5755/j01.itc.50.2.25583</b>	<b>Research on Data Currency Rule and Quality Evaluation</b>	
	Received 2020/03/28	Accepted after revision 2021/04/26
	 <a href="http://dx.doi.org/10.5755/j01.itc.50.2.25583">http://dx.doi.org/10.5755/j01.itc.50.2.25583</a>	

**HOW TO CITE:** Duan, X., Guo, B., Shen, Y., Shen, Y., Dong, X., Zhang, H. (2021). Research on Data Currency Rule and Quality Evaluation. *Information Technology and Control*, 50(2), 247-263. <https://doi.org/10.5755/j01.itc.50.2.25583>

# Research on Data Currency Rule and Quality Evaluation

## Xuliang Duan

College of Computer Science; Sichuan University; No.24 South Section 1, Yihuan Road, Chengdu 610065, China; College of Information Engineering; Sichuan Agricultural University; No.46, Xinkang Road, Yaan 625014, China; phone: +8615008305394; e-mail: duanxuliang@sicau.edu.cn

## Bing Guo

College of Computer Science; Sichuan University; No.24 South Section 1, Yihuan Road, Chengdu 610065, China; phone: +8613980664852; e-mail: guobing@scu.edu.cn

## Yan Shen

School of Computer Science; Chengdu University of Information Technology; No.24 Block 1, Xuefu Road, Chengdu 610225, China; e-mail: sheny@cuit.edu.cn

## Yuncheng Shen, Xiangqian Dong, Hong Zhang

College of Computer Science; Sichuan University; No.24 South Section 1, Yihuan Road, Chengdu 610065, China; emails: {2014323040012, 2014323040006, 2014323040021}@stu.scu.edu.cn

**Corresponding authors:** guobing@scu.edu.cn, sheny@cuit.edu.cn

Data currency is a temporal reference of data, it reflects the degree to which the data is current with the world it models. Currency rule is a formal rule extracted from the data set and reflecting the currency order of the data tuples, it can be used for both data repairing and currency quality evaluation. Based on the research of data currency repairing, the basic form of currency rule is extended, and parallel rule extraction and update algorithms are proposed to meet the requirement of running on dynamic data sets. Besides, four data currency quality evaluation models are proposed and verified by experiments. The performance test show that the efficiency of parallel algorithms is significantly improved, the rules compliance mean(CM2) model based on extended currency rule has the highest average precision. The extended currency rules not only improve the efficiency and adaptability, but also provide more valuable features for data quality evaluation.

**KEYWORDS:** data currency, currency rule, data quality, quality evaluation, parallel algorithm.

## 1. Introduction

Currency is an important feature of data, it is a temporal reference that reflects the degree to which the data is current with the world it models. In data mining, data analysis, and data value-added applications, accurate data currency determines the reliability of data analysis results such as time series analysis, correlation, and recommendation. Scholars carried out research on data quality through direct observation, social investigation, theoretical derivation, etc., and obtained the characteristics that have great influence on data availability: accuracy, completeness, consistency, currency and entity identity [5]. Data quality is one of the key factors to determine the success or failure of a corporation, incorrect or incomplete data will jeopardize a corporation's ability in decision-making and implementation [2, 14]. Data quality reflects the availability and value of the data. Sargent [23] defined data quality as the ability to make all data meet practical needs, Wang and Strong [27] proposed the idea of "data quality is determined by whether these data could be applicable to the context and be suitable for data consumers." In 2002, the expert report pointed out that in the field of business, at least 2% commercial data is obsolete every month due to changes in customer information [7, 8]. There are a lot of time-disrupted data in our data sets, if we can't identify which one is „latest“, data queries may return incorrect results, and data analysis may lead to ambiguous conclusions, followed by data quality degradation and data value reduction.

In the era of big data and artificial intelligence, personal big data as a valuable asset is growing exponentially. As our various types of data are decentralized in various platforms and systems, the data quality problems caused by obsolescence and inaccurate currency become more and more serious [13, 28]. Personal data is a typical kind of dynamic data. The data reflecting the status of people's work and life is constantly changing with time, and the changing feature is also the biggest challenge in the data cleaning process. In personal data banking mode, data derives from different systems and platforms. The time attributes of these multi-source heterogeneous data are often inaccurate, which brings great challenges to data quality and data value [29]. For some attributes of the data, different times correspond to different values or different states, such as a person's degree

changes, marital status changes, etc. If the timestamp is incomplete or inaccurate, the order of the records cannot be determined which will brings great difficulties in data analysis and value-added application.

Take the following student's course selection records as an example, shown in Table 1, Eve's Database semester is missing, it is possible for us to repair it according to certain rules. For Alice and Bob, the following 3 rules „C Programming→Data Structure“, „C Programming→Database“, „Data Structure→Database“ all exist in their course selection sequence. Therefore, although Eve's Database semester is unknown, but as Eve has chosen a „Data Structure“ in his second semester, we can infer that the semester of Eve's „Database“ will not be earlier than that of „Data Structure“ according to Alice and Bob's rule „Data Structure→Database“. Although it is not certain whether the semester is 3 or 4, we know that it has a high probability of being greater than 2, so we can determine the order of the two records of Eve.

**Table 1**  
Student's Course Selection Tuples

tid	eid	name	course	semester
t1	S1	Alice	C Programming	1
t2	S1	Alice	Data Structure	2
t3	S1	Alice	Database	4
t4	S2	Bob	C Programming	1
t5	S2	Bob	Data Structure	2
t6	S2	Bob	Database	3
t7	S3	Eve	Database	NULL
t8	S3	Eve	Data Structure	2

This paper extended the basic form of currency rule, the extended rules can be updated incrementally on dynamic data sets, and the new added path length attribute can provide more effective information for currency quality evaluation. On this basis, parallel extraction and merging algorithms for currency rules are proposed, experimental tests show that the parallel algorithms are available and effective. Besides, four data currency quality evaluation models are proposed and verified by experiments.

## 2. Related Works

### 2.1. Literature Review

There are many reasons for uncertain data, but overall the factors can be attributed to objective limitations and subjective intentions. Objective limitations include environmental impact, accuracy of acquisition equipment, transmission errors and accidents caused by missing or inaccurate data; subjective intentions usually refer to confuse or disrupt the original accurate data for privacy and security reasons. Research on data repairing mainly focuses on data inconsistency, inaccuracy, incompleteness, etc. Researchers and engineers have proposed a series of data repairing methods for these problems. Since the 1980s, probabilistic database related research on deterministic data has been carried out. This kind of research work introduces uncertainty into the relational data model. The research field covers model definition, pre-processing and integration, storage and indexing, query and analytical and other aspects of basic and applied research [12, 31]. Early data currency researches mainly focus on the query problems of uncertain databases, try to obtain the most accurate results on the dataset missing or with inaccurate timestamps. The main work involves the construction of temporal database models, the definition of query language, etc. [17, 22].

About the research on data currency repairing, solutions can be roughly divided into two types: semantic-based methods and statistical-based methods. Semantic-based methods find data errors and fix them based on domain expert knowledge, or functional dependencies, conditional dependencies, and time-dependent dependencies [1, 11]; Statistical-based approach aims to obtain the most possible correct repair strategy by analysing data laws, such as possible world models and probabilistic databases [16, 30].

Although the currency of data is facing problems, it is not hopeless. Around 2011, Fan et al. proposed the currency-repairing method based on data semantics. Under the assumption that one entity may have multiple tuples in datasets, Fan et al. conducted in-depth research on the fields of model for data currency, reasoning about data currency and currency preserving copy functions, etc., and formalized the related definition and concepts, and promoted basic research in this field [9, 10]. On the basis of this work, a series of fruit-

ful researches have promoted the research progress of data currency-repairing in theory and practice. Fan et al. [11] inferred the available time information of the data according to the currency order of the data, determined the latest value of the data according to the currency rules and constant conditional dependency function, and solved the inconsistency of different tuple attributes of the same entity to a certain extent.

From 2012 to 2016, Li et al. carried out researches on data currency and currency evaluation, proposed a series of solutions to the problems of currency determination and currency repairing [18-20]. They investigated the methods of currency evaluation with redundant records and currency constraints, proposed the definition of currency graph, and presented the methods of evaluating data currency relative to queries and users using currency graphs [19]; They also proposed a new class of rules which combining the quality rules and statistical techniques to improve data currency, domain knowledge can be directly expressed by the antecedents and consequents of rules, and the statistical information can be described by the distribution table of each rule [18, 20]. Ding et al. improved the currency determination and updating efficiency by constructing entities query B-Tree and static-dynamic link lists on the dynamic dataset [3, 4].

The recovery of data timestamps is a research hotspot. If the data timestamp is missing, it is difficult to perform exact repair; while, according to some rules, it is a feasible strategy to restore the data sequence and determine the interval range of missing timestamps. Song et al. proposed a method for data timestamps repairing using currency constraints [25]. The method adopts the minimizing modification principle to make the event meet the currency constraint condition. On the one hand, it can recover the time range of the event that missing timestamp, on the other hand, it can also be used for detecting and repairing the event that violates the currency constraint. In fact, in most data analysis and applications, the dependence on the data sequence relationship is more important than the precise time. A series of algorithms such as time series, association [24], recommendation, etc. are basically based on the „sequence“ but not accuracy timestamp for analysis and processing. Therefore, the repair of the time sequence of missing timestamp data can satisfy the requirements of „sequence“ of most analysis applications, and can also determine

that the „new and old“ records meet the time-sensitive query requirements, which is beneficial to the improvement of data quality and data value.

Based on the requirements of repairing data currency order, Duan et al. proposed a data currency rule model that does not require domain knowledge [6]. The algorithm extracts the currency rules and their support value by scanning the state changes of an attribute value of the data set. The currency rules can be used to repair data that missing currency order, and also provides feasibility solution for the quality evaluation of data currency. Liang et al. analysed and mined the students' course selection data set based on the currency rules and decision trees [21]. They evaluated each student's course selection data based on the mean value of the rules' supports, and found that data with poor currency quality often correspond to some abnormal situations of students, which can be used for abnormal warning in teaching management.

## 2.2. Data Currency Rule

**Currency rules for attribute values.** The currency rules defined in this paper refer to some regularities of certain attributes of the same entity over different time periods. Let the relational database schema  $R = (EID, A_1, \dots, A_n)$ , where EID is the entity identifier, different tuples with the same EID correspond to the same entity. EID can be generated by the entity identification technology [15, 26];  $A_i$  is the  $i$ -th attribute of the schema  $R$ , and its value range is  $dom(A_i)$

$$\forall t_1, t_2 \in R, (t_1[EID] = t_2[EID] \wedge t_1 \prec_{A_i} t_2) \rightarrow t_1 \prec_{A_i} t_2.$$

That is, for the tuples  $t_1$  and  $t_2$  of the same entity in the relationship  $R$ , if  $t_2$  is "newer" than  $t_1$ , the attribute  $A_i$  of the tuple  $t_2$  is newer than  $t_1$ .

Here, we assume that the data has been cleaned and it is consistent, does not violate functional dependencies and conditional dependencies, that is, there is no new attribute  $A_i$  in  $t_1$  than  $t_2$ , then the following relationship still holds:

$$\forall t_1, t_2 \in R, (t_1[EID] = t_2[EID] \wedge t_1 \prec_{A_i} t_2) \rightarrow t_1 \prec t_2.$$

That is, for two tuples  $t_1$  and  $t_2$  of the same entity, if  $A_i$  satisfies the currency rule, and the  $A_i$  of  $t_2$  is newer than  $t_1$ , the tuple  $t_2$  is newer than  $t_1$ .

For example, our age increases with time. If two records of a person have the age attribute, respectively, 23 and 25, then the record with age 25 is obviously newer. It is possible to use such attribute values that reflect the records sequence to repair the data. Similarly, some state changes can also reflect the sequence of records. For example, in a person's records at different periods, the status of a degree may be *Bachelor*, *Master*, or *Doctor*, and the marital status may be *unmarried*, *married* and *divorced*. We can confirm that for a person, his record with *Doctor* degree is newer than the record with *Master* degree, and record with *Married* marital status is newer than that of *Unmarried*. For states, we cannot directly compare the state values, but it is feasible to use state transitions to indicate their sequential relationship.

**Currency rules for attribute states.** On relationship  $R$ , all the non-repetitive state values of the attribute  $A_i$  are a finite set, and the currency rules state diagram are expressed by a directed graph  $G(V, E)$ , wherein the vertex set  $V$  represents the finite set of attribute values (states), and the directed edge set  $E$  represents the direction of state transition with chronological order. For the two tuples  $t_1$  and  $t_2$  of the same entity on the relationship  $R$ , the values of the attributes  $A_i$  are  $v_1, v_2$ , and  $v_1$  and  $v_2$  are the state nodes in the graph  $G$ . If  $t_2$  is newer than  $t_1$ , for the nodes  $v_1$  to  $v_2$  is reachable and meanwhile  $v_2$  to  $v_1$  is unreachable, we call  $v_1 \rightarrow v_2$  is a state currency rule, which is expressed as follows:

$$\begin{aligned} \forall t_1, t_2 \in R, t_1[EID] = t_2[EID], t_1[A_i] = v_1, \\ t_2[A_i] = v_2, v_1 \in G, v_2 \in G \\ (v_1 \rightarrow v_2 \wedge \neg v_2 \rightarrow v_1) \rightarrow t_1 \prec t_2 \end{aligned} \quad (1)$$

Similarly, on a consistent data set, we can determine:

$$\begin{aligned} \forall t_1, t_2 \in R, t_1[EID] = t_2[EID], t_1[A_i] = v_1, \\ t_2[A_i] = v_2, v_1 \in G, v_2 \in G \\ t_1 \prec t_2 \rightarrow (v_1 \rightarrow v_2 \wedge \neg v_2 \rightarrow v_1) \end{aligned} \quad (2)$$

The transition of attribute states can be represented by state diagrams. Figure 1 shows two sample state transition diagrams. It should be noted that in the degree state transition diagram, *Bachelor* to *Master*, *Master* to *Doctor* are all irreversible transition, and even if the intermediate state *Master* is ignored, the *Bachelor* to *Doctor* transition is also irreversible. Si-

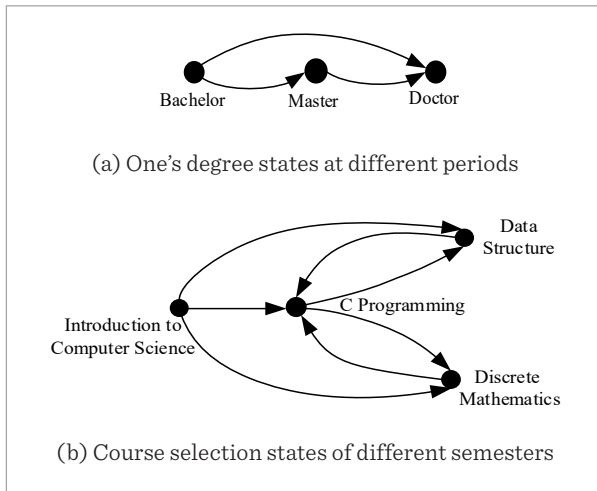
milar situation exists in the course election transition diagram. So, we have the following inference:

*A non-binary currency rule can be expressed as several binary currency rules. If the number of rule nodes is  $N$ , the maximum number of unique binary rules is:*

$$MaxBinaryRules = \sum_{i=1}^{N-1} (N - i) = \frac{N \times (N - 1)}{2}. \quad (3)$$

For example, „ $a \rightarrow b \rightarrow c \rightarrow d$ “ can be expressed as six unique binary currency rules: „ $a \rightarrow b$ “, „ $a \rightarrow c$ “, „ $a \rightarrow d$ “, „ $b \rightarrow c$ “, „ $b \rightarrow d$ “, „ $c \rightarrow d$ “.

**Figure 1**  
Degree and course selection state transition diagrams



**Support of the currency rule.** Support refers to the proportion of entities that satisfying a certain rule. The support is defined as follows:

$$S_r = \frac{|O(r)|}{|O(r)| + |V(r)|} \times f(r), \quad (4)$$

where  $S_r$  represents the support of a rule  $r$ ,  $O(r)$  represents the set of entities satisfying the rule  $r$ ,  $|O(r)|$  represents the number of entities in the set, and  $V(r)$  represents the set of entities that violate the rule,  $|V(r)|$  is the number of entities that violate the rule  $r$ .

$f(r)$  is an intensity function that could be intuitively described as: the more repetitions of a rule, the higher of the value. It should be noted that, refer to the de-

inition of rule support, if there is one and only one directed edge from a certain state node  $v_1$  to  $v_2$ , but  $v_2$  to  $v_1$  are unreachable, the support of the rule  $v_1 \rightarrow v_2$  is 100%, but this rule may be a special case with low repeatability, poor representation, and little value. The intensity function should be defined according to the actual features of the data. For the definition of the intensity function  $f(r)$  in this paper, see the following Section 3.1.

### 3. Parallel Currency Rule Algorithms

#### 3.1. Extending for the Basic Currency Rule

Duan et al. proposed the basic form of the currency rule is  $R = \{rule, support\}$ , and the rules are extracted by scanning all the records of a data set [6]. After extraction, if new records are added to the data set, it is necessary to scan the entire new data set to update previous currency rule set. Moreover, the currency rules updating can only be performed on a single node, the cost is high for dynamic, large-scale data set. In order to achieve the parallelization of the algorithms, we improved and extended the form of basic currency rule. The extended currency rule extraction and updating algorithms can not only run on multiple nodes in parallel, but also incrementally update the currency rule set on a dynamic data set.

The extended currency rule retains more information in the form of:

$$R = \{rule, obey, violate, length\}.$$

The *obey* is the number of entities that satisfy the rule in the data set. The *violate* indicates the number of entities that violate the rule in the data set. The *length* represents the average number of edges between the start and the end state in the rule. After expansion, the support of a rule can be dynamically calculated when needed by *obey* and *violation* values:

$$S_r = \frac{obey}{obey + violate} \times f(r). \quad (5)$$

In this paper, we take the following intensity function  $f(r)$ :

$$f(r) = \frac{|obey - violate|}{c + |obey - violate|}, \quad (6)$$



where  $c$  is a constant, which can be regarded as a threshold value for filtering low-frequency rules. For example, we take  $c = 10$  in the experiment, when the difference between obedience and violation is 10, the intensity value is 0.5, the larger the difference, the closer the intensity value is to 1.0.

The expanded currency rules have the following important properties:

**Property 1: The expanded rules are reversible.** The inverse form of a rule  $\{a \rightarrow b, obey, violate, length\}$  can be expressed as  $\{b \rightarrow a, violate, obey, length\}$ .

**Property 2: The expanded rules are additive.** The two or more same rules extracted from different data sets can be added into one rule. For example, the following two rules  $R1 = \{a \rightarrow b, o_1, v_1, len_1\}$  and  $R2 = \{a \rightarrow b, o_2, v_2, len_2\}$  can be merged into:

$$R = \{a \rightarrow b, o_1 + o_2, v_1 + v_2, len\}$$

$$len = \frac{(o_1 + v_1) \times len_1 + (o_2 + v_2) \times len_2}{o_1 + v_1 + o_2 + v_2} \quad (7)$$

The two important properties of rule reversibility and additivity are important foundations for parallelization of related algorithms.

### 3.2. Currency Rule Extraction Algorithm

Currency rule extraction algorithm is shown in *Algorithm 1*. The input data set consists of records from multiple entities, each entity has multiple records, and each record tuple has an order label attribute. For example, the above Table 1 contains 8 tuples of 3 entities (3 students), and the *semester* can be regarded as the order label. The input data set should be pre-processed, cleaned and consistent data.

In Algorithm 1, the  $rule'$  in line 6 is the inverse rule of the current currency rule. Lines 7 to 13 respectively update the rule and its inverse rule in rule set. The rule update process in Algorithm 1 is shown in *Algorithm 2*. This algorithm updates the existing rule in a currency rule set according to the attribute values of the new rule.

**Example 1. Currency Rule Extraction.** The calculation process is illustrated by the following example. Suppose there are now three entities,  $e_1, e_2, e_3$ , and each entity has multiple records. An attribute of the records of the entities  $e_1, e_2$  all have a sequence „ $a \rightarrow b \rightarrow c \rightarrow d$ “, entity  $e_3$  have a sequence „ $a \rightarrow c \rightarrow b \rightarrow d$ “.

---

#### Algorithm 1. Currency Rule Extraction Algorithm

**Input:** A data set containing multiple entities for rule extraction;

**Output:** Currency rule set  $CRS$  of the attribute  $A_k$

```

1:  for each  $e \in E$  do
2:     $T \leftarrow$  select all tuples of  $e$  from dataset order by
      attribute  $A_{order}$  ascending
       $N \leftarrow$  tuples count  $|T|$ 
3:    for each  $i$  from 1 to  $N-1$  do
4:      for each  $j$  from  $i+1$  to  $N$  do
5:         $rule \leftarrow$  “ $T_i[A_k] \rightarrow T_j[A_k]$ “,  $obey \leftarrow 1, violate \leftarrow 0,$ 
           $length \leftarrow T_j[A_{order}] - T_i[A_{order}]$ 
6:         $rule' \leftarrow$  “ $T_j[A_k] \rightarrow T_i[A_k]$ “,  $obey' \leftarrow 0, violate' \leftarrow 1,$ 
           $length' \leftarrow T_i[A_{order}] - T_j[A_{order}]$ 
7:        if  $CRS$  contains  $rule$  then
8:          UpdateRule( $rule, obey, violate, length$ )
9:        else if  $CRS$  contains inversed  $rule'$  then
10:         UpdateRule( $rule', obey', violate', length'$ )
11:        else AddRule( $rule, obey, violate, length$ )
12:        end if
13:      end for
14:    end for
15:  end for
16:  return  $CRS$ 

```

---

#### Algorithm 2. Currency Rules Update Algorithm

**Input:** Currency rule set  $CRS$ , and rule in the form of  $\{rule, obey, violate, length\}$ .

**Output:** The updated currency rule set  $CRS$

```

1:  if  $CRS$  contains  $rule$  then
2:     $\{rule, ro, rv, rl\} \leftarrow$  select the currency rule from
       $CRS$  where rule name is  $rule$ 
3:     $rl \leftarrow [(ro + rv) * rl + (obey + violate) * length] / (ro +$ 
       $rv + obey + violate)$ 
4:     $ro \leftarrow ro + obey$ 
5:     $rv \leftarrow rv + violate$ 
6:    update rule with new values  $\{rule, ro, rv, rl\}$ 
7:  end if
8:  return  $CRS$ 

```

---

According to Algorithm 1, the currency rules of entity  $e_1$  are first extracted. All records of entity  $e_1$  are sorted in ascending order of time labels, and the state transition rules of a certain attribute are extracted. The currency rules extracted from the state sequence „ $a \rightarrow b \rightarrow c \rightarrow d$ “ are listed in Table 2. The right half of the table is the corresponding inverse rules, which can be deduced from the positive rules, and generally, it is not necessary to store the inverse rules.

**Table 2**  
Currency Rules Extracted from All Records of Entity  $e_1$

$R$	$o$	$v$	$len$	$R'$	$o'$	$v'$	$len'$
$a \rightarrow b$	1	0	1.00	$b \rightarrow a$	0	1	1.00
$a \rightarrow c$	1	0	2.00	$c \rightarrow a$	0	1	2.00
$a \rightarrow d$	1	0	3.00	$d \rightarrow a$	0	1	3.00
$b \rightarrow c$	1	0	1.00	$c \rightarrow b$	0	1	1.00
$b \rightarrow d$	1	0	2.00	$d \rightarrow b$	0	1	2.00
$c \rightarrow d$	1	0	1.00	$d \rightarrow c$	0	1	1.00

Next, the records of the entity  $e_2$  are scanned to extract and update the rules. According to the principle of additivity and Algorithm 2, the updated currency rules are shown in Table 3.

**Table 3**  
Currency Rules Extracted from All Records of Entity  $e_1$  and  $e_2$

$R$	$o$	$v$	$len$	$R'$	$o'$	$v'$	$len'$
$a \rightarrow b$	2	0	1.00	$b \rightarrow a$	0	2	1.00
$a \rightarrow c$	2	0	2.00	$c \rightarrow a$	0	2	2.00
$a \rightarrow d$	2	0	3.00	$d \rightarrow a$	0	2	3.00
$b \rightarrow c$	2	0	1.00	$c \rightarrow b$	0	2	1.00
$b \rightarrow d$	2	0	2.00	$d \rightarrow b$	0	2	2.00
$c \rightarrow d$	2	0	1.00	$d \rightarrow c$	0	2	1.00

Finally, scan all records of entity  $e_3$  and update the currency rules. The currency rules extracted from all records of the three entities  $e_1, e_2, e_3$  are shown in Table 4. It should be noted that the new rule „ $c \rightarrow b$ “ does not exist in the rule set, but its inverse rule „ $b \rightarrow c$ “ exists, according to the principle of reversibility of the rule, we can update the rule „ $b \rightarrow c$ “ instead.

**Table 4**  
Currency Rules Extracted from All Records of 3 Entities

$R$	$o$	$v$	$len$	$R'$	$o'$	$v'$	$len'$
$a \rightarrow b$	3	0	1.33	$b \rightarrow a$	0	3	1.33
$a \rightarrow c$	3	0	1.67	$c \rightarrow a$	0	3	1.67
$a \rightarrow d$	3	0	3.00	$d \rightarrow a$	0	3	3.00
<b><math>b \rightarrow c</math></b>	<b>2</b>	<b>1</b>	<b>1.00</b>	<b><math>c \rightarrow b</math></b>	<b>1</b>	<b>2</b>	<b>1.00</b>
$b \rightarrow d$	3	0	1.67	$d \rightarrow b$	0	3	1.67
$c \rightarrow d$	3	0	1.33	$d \rightarrow c$	0	3	1.33

### 3.3. Rule Sets Merging Algorithm

The currency rule extraction algorithm can run in parallel. A possible parallel solution is to divide a large data set into multiple small ones and distribute them on different nodes, each node independently runs currency rule extraction algorithm, finally, the *Rule Sets Merging Algorithm* merges these rule sets extracted from small data sets into a complete currency rule set. The merge algorithm solves the problem of parallel rule extraction, and it also supports incrementally updating the rule set on dynamic increasing data. Algorithm 3 shows the merging process:

**Algorithm 3. Rule Sets Merging Algorithm**

**Input:** Two Currency Rule Sets,  $RuleSet_1$  and  $RuleSet_2$

**Output:** The merged  $RuleSet_1$

- 1: **for** each *rule* in  $RuleSet_2$  **do**
- 2:   **if**  $RuleSet_1$  contains *rule* **then**  
       Formalize the *rule* into  $\{rule, obey, violate, length\}$
- 3:     UpdateRule(*rule*, *obey*, *violate*, *length*) for  $RuleSet_1$
- 4:   **else if**  $RuleSet_1$  contains inverse rule *rule'* **then**
- 5:     UpdateRule(*rule'*, *violate*, *obey*, *length*) for  $RuleSet_1$
- 6:   **else** AddRule(*rule*, *obey*, *violate*, *length*) to  $RuleSet_1$
- 7:   **end if**
- 8: **end for**
- 9: **return**  $RuleSet_1$

**Example 2. Incremental Update of Currency Rules.**

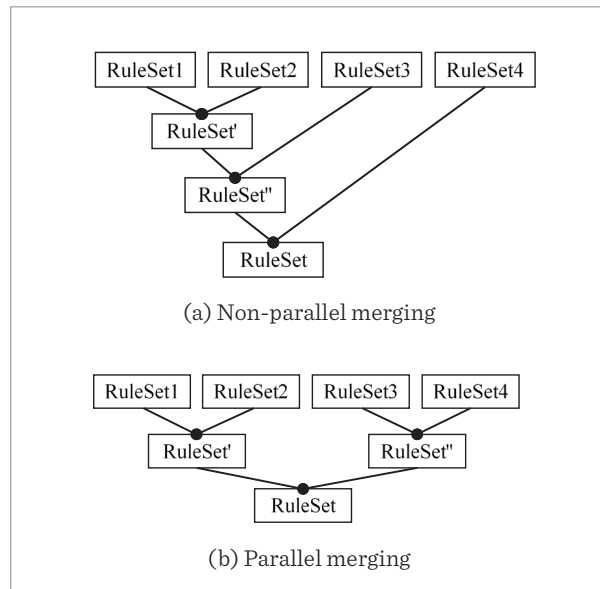
The algorithm supports incremental updating of currency rules on dynamic data sets. In Example 1, the currency rules have been extracted, and the *obey, vio-*

late, length values of each rule have been processed. Assuming that some new tuples are added to the original data set, there is no need to scan the entire set, the rule set only needs to be incrementally updated. Assuming that the currency rule set extracted from the new added tuples are  $\{\{a \rightarrow e, 3, 0, 2\}, \{a \rightarrow b, 5, 2, 2\}, \{d \rightarrow c, 10, 1, 2\}\}$ . For each rule in the new rule set, merge it into the existing rule set according to the process described in Algorithm 3, for this example, three rules “ $a \rightarrow e$ ”, “ $a \rightarrow b$ ”, “ $d \rightarrow c$ ” need to be added or updated. The merged rule set are shown in Table 5.

**Table 5**  
The Merged Two Rule Sets

$R$	$o$	$v$	$len$	$R'$	$o'$	$v'$	$len'$
$a \rightarrow b$	8	2	1.80	$b \rightarrow a$	2	8	1.80
$a \rightarrow c$	3	0	1.67	$c \rightarrow a$	0	3	1.67
$a \rightarrow d$	3	0	3.00	$d \rightarrow a$	0	3	3.00
$b \rightarrow c$	2	1	1.00	$c \rightarrow b$	1	2	1.00
$b \rightarrow d$	3	0	1.67	$d \rightarrow b$	0	3	1.67
$c \rightarrow d$	4	10	1.86	$d \rightarrow c$	10	4	1.86
$a \rightarrow e$	3	0	2	$e \rightarrow a$	0	3	2

**Figure 2**  
Two rule sets merging strategies



When merging rule sets, two strategies can be used. The first strategy is to merge the 2, 3, 4, ...,  $n$  rule set into the first set. After the last set is merged, the first set is a complete rule set. This strategy can't be executed in parallel, and the time complexity is  $O(n-1)$ ; The second strategy, rule sets merging can be run in parallel on different nodes and finally be merged into a complete rule set, the time complexity is  $O(\log(n))$ . The two strategies diagrams are as shown in Figure 2.

### 4. Models for Data Currency Quality Evaluation

The currency quality evaluation refers to a quantitative evaluation of the degree to which the state transition of an attribute of all tuples of an entity is consistent with the currency rules. If the quality of each entity in a data set is high, the currency quality of the data set is also high. To evaluate the currency quality of an entity  $e$ , first query all the tuples of  $e$  from the data set, order the tuples by order label ascending, and then extract all the currency rules from the sorted tuples. Finally, use a specific model to evaluate the consistency of the extracted rules with the currency rule set.

A sample currency rule set are shown in Table 6. The rules are all from the above Table 5 and we calculated the *intensity* and *support* of each rule according to the relevant currency rule definition. For demonstration purposes, the constant threshold in intensity function  $f(r)$  is set to  $c = 1.0$ .

**Valid Rule.** Here, we consider a rule with *support*  $> 0.5$  as a **valid rule**. On the one hand, the number of entities obeying the rule must be greater than the number of violations, on the other hand, the rule must appear more than a certain frequency. Of course, it can also be determined that a rule with a *support*  $> 0.6$  is the **valid rule**.

**Consistent Rule.** If the current rule exists in the rule set and it is a **valid rule**, we consider the current rule to be a **consistent rule**; if the inverse rule of the current rule exists in the rule set and it is a **valid rule**, we consider the current rule to be an **inconsistent rule**. For example, “ $a \rightarrow b$ ” is a consistent rule, “ $b \rightarrow a$ ” is an inconsistent rule; “ $d \rightarrow c$ ” is a consistent rule and “ $c \rightarrow d$ ” is an inconsistent rule.



**Table 6**  
A Sample Rule Sets

<i>R</i>	<i>o</i>	<i>v</i>	<i>intensity</i>	<i>support</i>	<i>len</i>
a→b	8	2	0.86	0.69	1.80
a→c	3	0	0.75	0.75	1.67
a→d	3	0	0.75	0.75	3.00
b→c	2	1	0.5	0.33	1.00
b→d	3	0	0.75	0.75	1.67
c→d	4	10	0.86	0.24	1.86
a→e	3	0	0.75	0.75	2.00
<i>R'</i>	<i>o'</i>	<i>v'</i>	<i>intensity'</i>	<i>support'</i>	<i>len'</i>
b→a	2	8	0.86	0.17	1.80
c→a	0	3	0.75	0	1.67
d→a	0	3	0.75	0	3.00
c→b	1	2	0.50	0.17	1.00
d→b	0	3	0.75	0	1.67
d→c	10	4	0.86	0.61	1.86
e→a	0	3	0.75	0	2.00

In this paper, we construct four quantitative evaluation models, analyse and discuss the evaluation effect of each model. Suppose there are now two entities  $e_5, e_6$  need to be evaluated, for a specific attribute, the state transition sequences are separately “c→b→d” and “a→b→d”, and the extracted rules are respectively  $e_5\{c→b, c→d, b→d\}$  and  $e_6\{a→b, a→d, b→d\}$ . In the following, the definition of the evaluation models will be explained with intuitive examples.

**4.1. Consistent Rules Ratio**

**Consistent Rules Ratio (CR).** Consistent rules ratio is the degree to which all the rules extracted from all records of an entity  $e$  are consistent with the valid rules in currency rule set. The larger the ratio of the number of consistent rules, the better the entity’s data currency quality. The *CR* model is defined as:

$$CR = \frac{|ConsistentRules|}{|ConsistentRules| + |InconsistentRules|}, \tag{8}$$

where the  $|ConsistentRules|$  and  $|InconsistentRules|$  represent the number of consistent rules and the number of inconsistent rules, respectively.

Take the entities  $e_5\{c→b, c→d, b→d\}$  and  $e_6\{a→b, a→d, b→d\}$  as an example, for  $e_5, support(c→b)=0.17, support(c→d)=0.24$ , both “c→b”, “c→d” are not valid rule, only rule “b→d” with  $support(b→d)=0.75$  is a valid rule, so  $CR(e_5)=0.33$ . For entity  $e_6, “a→b”, “a→d”, “b→d”$  are all valid,  $CR(e_6)=1.00$ . Therefore, it can be considered that the rules from records of entity  $e_6$  are more consistent with the existing rules, and the currency quality of the entity  $e_6$  is better than  $e_5$ .

**4.2. Rules Support Mean**

**Rules Support Mean (SM).** Suppose  $n$  rules are extracted from all records of an entity  $e, s_i$  is the support value of rule  $i$ , use the mean of rules’ support as a model to evaluate the currency quality of entity  $e$ . The larger the *SM* value, the better the currency quality of the data. The support mean is defined as follows:

$$SM = \frac{\sum_{i=1}^n s_i \times flag_i}{n}, \tag{9}$$

$$flag = \begin{cases} 1, ConsistentRule \\ -1, InconsistentRule \end{cases} \tag{10}$$

In *SM* model, *flag* is used to indicate whether the current rule is a consistent rule. If it is, the *flag* is 1, if not, it is -1; If the current rule and its inverse rule are both not valid rule in the rule set, the *flag* and support value are both 0.

Take the above two entities  $e_5$  and  $e_6$  as examples, for the three rules of entity  $e_5$ , only “b→d” is a consistent rule, and “c→d” is a consistent rule, therefor,  $SM(e_5) = (0+0.75-0.61)/3=0.05$ ; Three rules of entity  $e_6$  are all consistent rules,  $SM(e_6) = (0.69+0.75+0.75)/3=0.73$ , so it can be considered that the currency quality of the entity  $e_6$  is better than  $e_5$ .

**4.3. Rules Compliance Mean**

**Rule Compliance.** Rule compliance reflects the degree of consistency between the current rule and its corresponding rule in the rule set. Rule compliance is defined as follow:

$$C = \frac{Len + RuleLen - |Len - RuleLen|}{Len + RuleLen} \times flag, \tag{11}$$

where *Len* is the path length of the current rule, *RuleLen* is the path length of corresponding valid rule in

rule set; if there is not a corresponding valid rule in rule set, let  $RuleLen = 0$ . When  $Len = 0$  and  $RuleLen = 0$ , let compliance  $C = 1$ . The  $flag$  indicates whether the current rule is a consistent rule, see formula (10). The value range of compliance is  $[-1, 1]$ .

**Rules Compliance Mean (CM).** Here we use the mean value of compliance of  $n$  rules extracted from multiple records of an entity as a model to evaluate the data quality. Two types of compliance mean models are defined, one does not consider the impact of rule support on the evaluation results, and the other one can reflect the impact of rule support on result, respectively  $CM1$  and  $CM2$ .

$$CM1 = \frac{\sum_{i=1}^n C_i \times flag_i}{n}, \quad (12)$$

$$CM2 = \frac{\sum_{i=1}^n C_i \times flag_i \times s_i}{n}. \quad (13)$$

Take the above two entities  $e_5(c \rightarrow b \rightarrow d)$  and  $e_6(a \rightarrow b \rightarrow d)$  as an example, the Compliance,  $CM1$  and  $CM2$  results are shown in Table 7. For entity  $e_5$ , after considering the influence of support, the  $CM2 > CM1$ , because the support of a consistent rule “ $b \rightarrow d$ ” is higher than the inconsistent rule “ $c \rightarrow d$ ”.

**Table 7**

Currency Quality Evaluation Result

Entity	Rule	Length	flag	Compliance	CM1	CM2
$e_5$	$c \rightarrow b$	1.0	0	0.00	-0.07	-0.01
	$c \rightarrow d$	2.0	-1	-0.96		
	$b \rightarrow d$	1.0	1	0.75		
$e_6$	$a \rightarrow b$	1.0	1	0.71	0.75	0.55
	$a \rightarrow d$	2.0	1	0.80		
	$b \rightarrow d$	1.0	1	0.75		

From the above results, we can consider that the rules from records of entity  $e_6$  is more consistent with the existing rules, and the currency quality of the entity  $e_6$  is better than that of  $e_5$ .

## 5. Experiment and Analysis

### 5.1. Experimental Platform and Data

The experimental platform is a host allocated from a hyper-converged server. The host is configured with Intel T7700 2.4GHz CPU, 20 cores, 16G RAM and Windows 10 operating system. The algorithms are all written in C#, running on .Net Framework 4.5. The test datasets and rules set are stored in MySQL 5.7 database.

The experimental data set is derived from the course elective data of students in a university from 2014 to 2019. The data consistency is high and there is no abnormal data. The data fields include the course name, class number, student number, and semester. Generally, compulsory courses are usually scheduled in fixed semesters according to the training program, which has strong time sequence feature, and there are no clear restrictions on elective courses. However, the course election of students in this university is more flexible, students can choose compulsory courses or elective courses for their own majors or other majors according to their own circumstances, and students can also take compulsory courses in advance or afterwards semesters. For example, a student may take a compulsory course that should be scheduled in seventh semester in his first semester, or one student may retake a course that has already been taken before, in order to get a higher score. Therefore, the elective data that students do not take courses according to the specified order in the training program can be regarded as noise data. Although there is no abnormal data in the data set, from the perspective of currency, there are still many uncertainties in the data.

The relevant terms mentioned in the experiment are explained as follows:

- State.** State refers to the node in the state diagram. In the experiment, we select the course name as the state, the same course name represents the same state, and different course names represent different states. The test courses election datasets have a total of more than 2,200 courses, which means that the number of nodes in the entire state diagram will not exceed the number of courses.
- Entity.** The entity in the experiment is the student ID. In the test dataset, an entity has multiple data records, that is, one student has multiple courses

elective records, and the records with the same student ID are the same student's elective data.

- 3 Currency rules of the state.** In the following description, it is referred to as the currency rule or rule. For the experimental datasets, the rule can be interpreted as the order of the students' courses election.

According to the experimental needs, the data is divided into two data sets:

Dataset 1, a total of 1247945 records, is the courses elective data from the first to the eighth semester of 18127 students of grade 2014 and 2015, with an average of about 69 records per student.

Dataset 2, a total of 8628 records, is the courses elective data of 132 students. The 132 students come from 4 classes of the same major of grade 2016. A total of 66 students from the 2 normal classes are marked as normal entities; the other 66 students in the other two classes with records of course retaking or major/program changing are marked as abnormal entities.

Dataset 1 is used to extract currency rules set, the dataset 1 and dataset 2 are used together for algorithm efficiency and parallel testing, dataset 2 is used to verify the validity of data currency quality evaluation.

## 5.2. Performance Testing for Rules Extraction

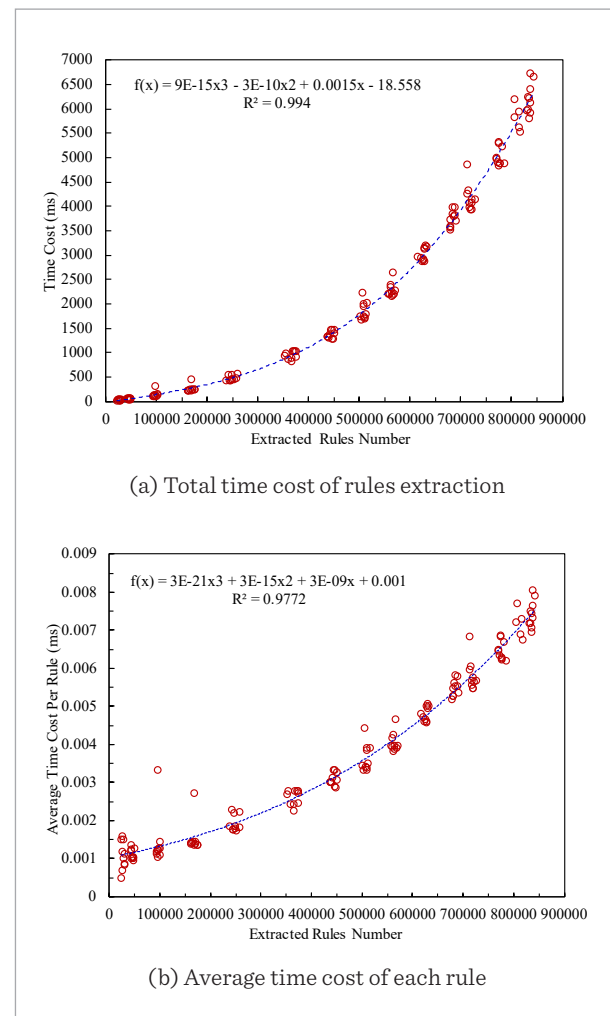
For the rule extraction performance test, currency rules are extracted from 15 entities sets of different sizes, test 10 rounds. Test entities are randomly selected from the Dataset 1 and Dataset 2, and the numbers of entities in the test data sets are sequentially incremented in a certain gradient. There are 10 entities in the first (smallest) set, and 2700 entities in the last (largest) set. Since the test entities are randomly selected, even if the number of entities of the test set is the same, the number of rules extracted from the set is generally different. Of all 10 rounds, the minimum number of rules for the set of 10 entities is 20482, and the maximum number for the set of 2700 entities is 852131.

### 5.2.1. Non-Parallel Test for Rules Extraction Algorithm

The results of 150 tests for all 10 rounds are shown in the Figure 3 below. The horizontal axis indicates the number of rules and the vertical axis indicates the time consumed (in milliseconds). Figure 3 (a) shows the time-consuming of extracting rules for each set

varies as the rules number increases. Figure 3 (b) shows the average time-consuming of each rule varies as the rules number increases. In the best case, 23403 rules are extracted per millisecond, and in the worst case, only 622 rules are extracted per millisecond. Due to computer storage and computational resource limitations, the rules extraction time consumption and the rules number do not show a linear growth trend of ideal conditions. In the Microsoft Excel, a trend line and a fitted polynomial function are added to the scatter plot, and it can be found that the time cost and the rules number shows a polynomial growth trend. A similar situation also occurs in the average time cost of each rule.

**Figure 3**  
Performance Test for Rules Extraction Algorithm

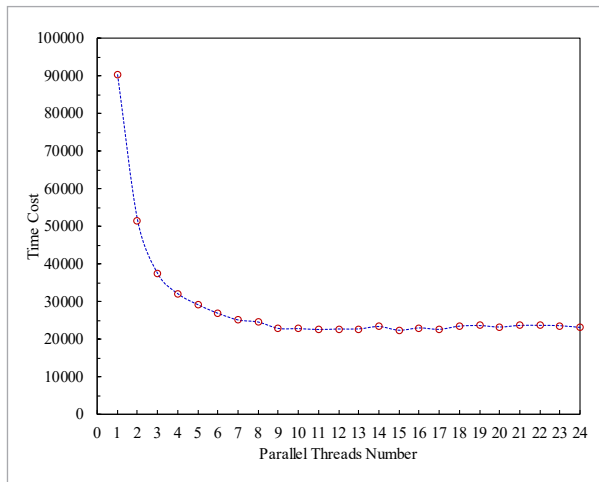


### 5.2.2. Parallel Test for Rules Extraction Algorithm

From the test dataset1 and dataset2, 1771292 records of 28076 entities are selected for doing parallel test. The total number of currency rules is 5573732. The test run 10 rounds, for each round, we set 1 to 24 parallel threads to test the algorithm performance under different parallel threads. Take the average as the result in the case of the same parallel threads number. The experimental results show that the parallelization has obvious improvement on the algorithm efficiency. For test task, single-threaded operation takes 90352.75 milliseconds, and when 15 threads are paralleled, it reaches a minimum of 22399.22 milliseconds, resulting in an efficiency increase of 75.20%. The experimental results also show that when the parallel threads number is greater than 10, due to server storage, disk IO, etc., continuing to increase parallel threads number does not continuously reduce time consumption. Test results are shown in Figure 4.

**Figure 4**

Parallel Test for Rules Extraction Algorithm



### 5.3. Performance Test for Rules Merging Algorithms

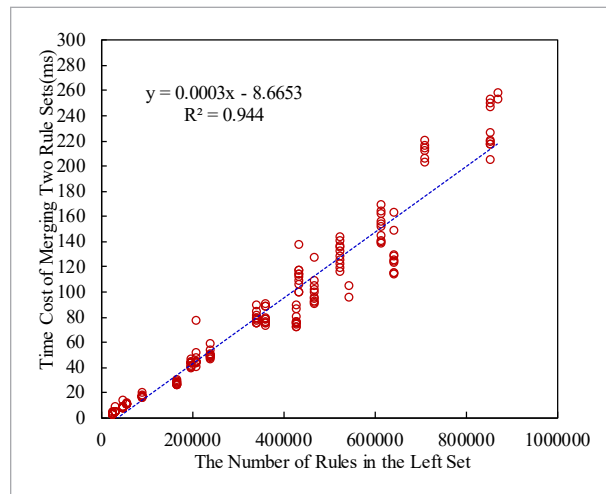
During the process of merging two rule sets, the left set  $R1$  and the right  $R2$ , it is necessary to traverse each rule in left set  $R1$  to check whether it exists in the rule set  $R2$ . Since  $R1$  and  $R2$  are both hash tables, the traversal  $R1$  algorithm is  $O(n)$  linear complexity, and check rule whether in  $R2$  is  $O(1)$  constant complexity. It can be seen that the merging cost is mainly determined by the size of the rule set  $R1$ .

### 5.3.1. Non-Parallel Test for Rule Sets Merging Algorithm

Performance test is carried out for 10 rounds. Each round of testing uses 30 entity sets, and each entity sets contains 10 to 5000 entities randomly selected from the data set under a certain gradient. Correspondingly, 30 rule sets are extracted from the 30 entity sets. Of the total 300 tests in 10 rounds, the rules number for the left rule sets ranges from 22317 to 867139, and for the right sets it is 12254 to 541369. There are 867,139 rules in the largest left set  $R1$ , and the corresponding right  $R2$  has 541369 rules. In the 10 rounds testing, the average time consuming of merging the two rule sets is 255.5ms. The results of the 10 rounds of testing are shown in Figure 5. It can be found that the merging time-consuming is linear with the scale of left rule set.

**Figure 5**

Performance Test for Rules Merging Algorithm



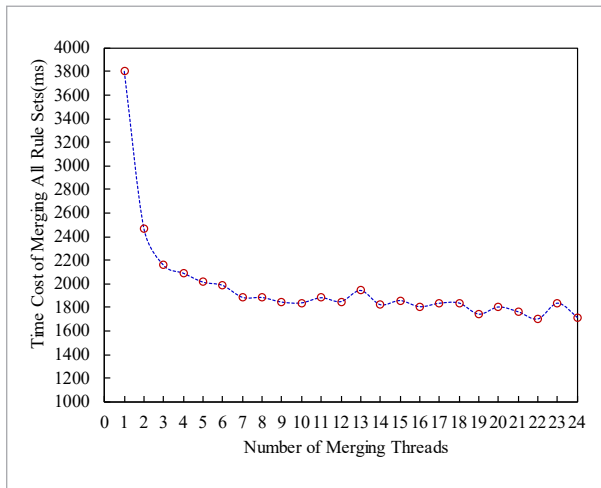
### 5.3.2. Parallel Test for Rule Sets Merging Algorithm

In parallel testing, for each round of testing, set 1 to 24 parallel threads to merge the 30 rule sets. Finally, count the data for 10 rounds of testing and calculate the average time cost based on the number of threads.

In the case of non-parallel (single-threaded), merging 30 rule sets takes an average of 3802 milliseconds, paralleling two threads takes 2466 milliseconds. A minimum of 1702 milliseconds is obtained when 22 threads are executed in parallel, brings efficiency im-

provement of 55.23%. Experiments show that the parallel algorithm is effective, and parallelization can significantly improve the merging efficiency. Since this experiment is performed on a single server, limited by hardware limitations and resource coordination between multiple threads, when the number of threads is greater than 10, the performance improvement is no longer obvious. The parallel performance results are shown in Figure 6.

**Figure 6**  
Parallel Test for Rules Merging Algorithm



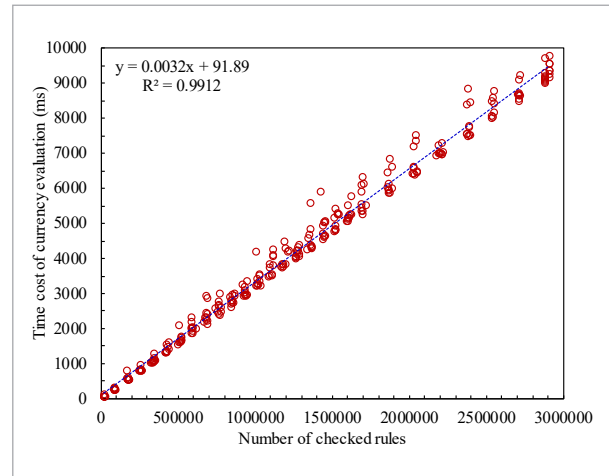
### 5.4. Performance Testing for Currency Evaluation

The data quality evaluation performance test is carried out for 10 rounds. Each round of testing uses 30 entity sets, and each entity sets contains 10 to 2000 entities randomly selected from the above dataset2. For each entity of each entity set, the currency quality is evaluated by the *Support Mean* model which mentioned in section 3.3. *Rules number* is the sum of rules extracted from all entities of an entity set, and the *Running time* is the total time to complete the quality evaluation of all entities in a set.

#### 5.4.1. Non-Parallel Test for Currency Evaluation

Figure 7 shows the non-parallel single-threaded test results. It can be seen that the time cost and the rules number show a linear growth trend. In the 10 rounds of tests, when the number of entities is 2000, the average rules number is 2906098, and the average time cost is 9745ms.

**Figure 7**  
Non-parallel test for currency evaluation

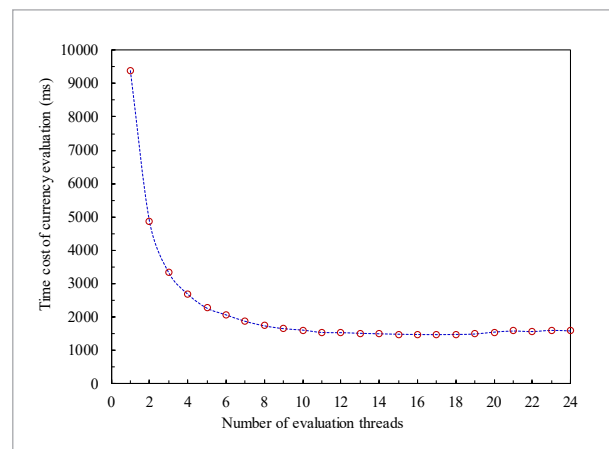


#### 5.4.2. Parallel Test for Currency Evaluation

In the parallel test, we randomly select a set of 2,000 entities from dataset 2 and need to evaluate the currency quality of each entity in the set. The test is run for 10 rounds, and each round runs 1-24 evaluation threads, and the results of 10 rounds are averaged.

There are 10 rounds of testing, and each round runs 1-24 threads, and the test results are averaged. When running in non-parallel, the average time consumption is 9379.33ms; when 17 threads are running in parallel, the average time consumption is the smallest, which is 1481.22ms, and the performance is improved by 85.86%. The results are shown in the following Figure 8.

**Figure 8**  
Parallel test for currency evaluation





### 5.5. Evaluation and Analysis of data Currency Quality Model

Evaluation and analysis of data currency quality model are based on the above dataset1 and dataset2. The currency rule set which contains 1654412 rules is extracted from the above dataset1. In dataset2, of the total 2329 entities, 2107 are marked as normal entities, and 222 are marked as abnormal entities. We need to use the rule set extracted from dataset1 to evaluate the currency quality of each entity in dataset2, and analyse the evaluation results of different models.

In the experiment, the recall and precision are used to evaluate different currency quality evaluation models. Calculate the precision of each model under the specific recall levels of 60%, 65%, 70%, 75%, 80%, 85%, 90%, and 95%. With the same recall level, the higher the precision, the better the evaluation model.

The recall and precision are defined as follow:

$$R = \frac{TP}{TP + FN}, P = \frac{TP}{TP + FP}, \quad (14)$$

where  $R$  is recall,  $P$  is precision,  $TP$  is the true positive,  $FP$  is the false positive and  $FN$  is false negative.

The precision and recall are often contradiction. Comprehensively consider the two indicators, we also use the weighted harmonic average of precision and recall, F-Measure, when  $P$  and  $R$  have the same weight, the result is  $F1$  score, defined as follows:

$$F1 = \frac{2 \times P \times R}{P + R}. \quad (15)$$

As there is no suitable public benchmark data set for our experiments, we prefer to use these  $R$ ,  $P$ ,  $F1$  indicators to evaluate the four proposed data quality evaluation models, rather than as absolute standards.

In the evaluation of currency quality models, we mainly evaluate their ability of finding abnormal entities, the experiment programs are as follows:

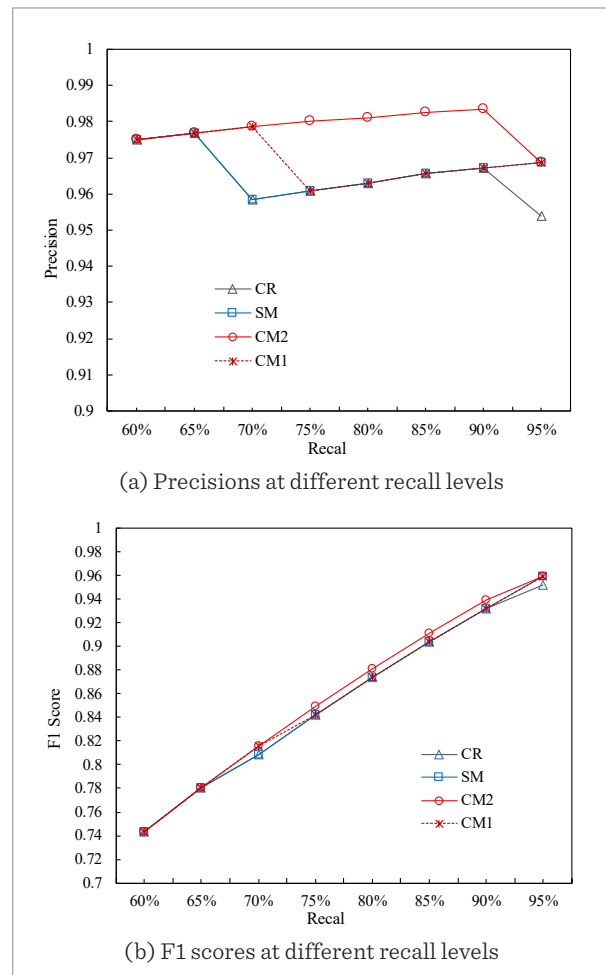
- 1 For each entity, extract the currency rules from all its records, and calculate the currency quality according to the evaluation models;
- 2 Sort all entities in ascending or descending order by the evaluated quality value;
- 3 Set the recall levels to 60%, 65%, ..., 95%, and calculate the precision of different recall levels. For example, in the identification of abnormal entities,

if the recall is set to 60%, the number of entities to be recalled is  $66 \times 60\% = 40$ . Check the sorted 66 entities order by model result ascending from front to back, count the normal and abnormal entities respectively, until the number of abnormal entities is 40. The count of abnormal entities is  $TP$ , and the count of normal entities is  $FP$ , according to the  $TP$  and  $FP$ , we can calculate the precision value of the specific recall level. The calculation of precision at other recall levels is the same.

The experiment compared the CR, SM, CM1 and CM2 evaluation models. In general, the precision of the four models are relatively high, and all of them can meet the needs of data currency quality evaluation. Precisions of each model at different recall levels are shown in Figure 9 (a).

Figure 9

Models' Precisions and F1 scores at different recall levels

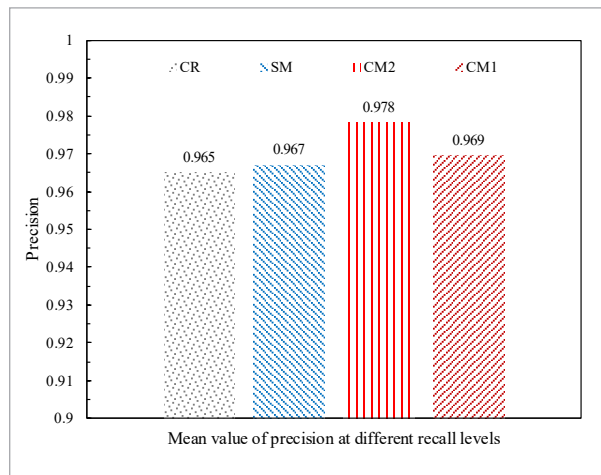


In general, the CM2 has the highest precision under the same recall level, precision reached a maximum of 0.9833 when the recall is 90%. The precisions at different recall levels of the CR, SM and CM1 models are very close. At the 70% recall level, CM1 is better than CR and SM; when the recall is 95%, the SM and CM1 are better than CR. When the recall is 95%, precision of CR model is the lowest at 0.9538.

The *F1* scores of each evaluation model at different recall levels are shown in Figure 9 (b). The trend of *F1* score of each model is basically consistent with the trend of precision, and the CM2 model performs best overall. The *F1* score of CM2 reached a maximum of 0.9695 when the recall is 95%.

**Figure 10**

Mean value of precisions at different recall level



The mean value of precisions at different recall level for 4 models are shown in Figure 10. Experimental results show that the CM2 model is generally optimal, followed by CM1 and SM, the CR model performs relatively inferior.

By analysing the definitions of four models, it can be found that the CR model only considers the direction consistence and has the least reference information; SM further uses support information on the basis of CR; CM1 uses the compliance degree of the rule, which is actually measures the consistency of the rules according to the path length feature. On the basis of CM1, the CM2 model references both the path length and the support features of the rules, making full use of the information of the extended currency rules, so the overall performance is best.

For data currency quality evaluation, first run the model on labelled data (for example, we marked entities as normal and abnormal), and then calculate the model mean of each type of data. In our experiment, using the CM2 model as an example, the average CM2 of entities marked as abnormal is 0.5761, and the average value of entities marked as normal is 0.6459. Therefore, if one's  $CM2 < 0.5761$ , it can be considered that its currency quality is very poor; while when another's  $CM2 > 0.6459$ , it can be considered that the entity's currency quality is very good. For more accurate quality grading, please refer to the recall-based method used in our experiment. In our experiment, the CM2 model under different recall levels, precisions, and model values are shown in Table 8. For example, we can determine that when an entity has a  $CM2 > 0.6477$ , there is a great possibility that it will also be an entity marked as normal, and the quality should be good.

**Table 8**

The CM2 Model Evaluation Result

Recall	Precision	CM2
60%	0.9750	0.5913
65%	0.9767	0.6044
70%	0.9787	0.6098
75%	0.9800	0.6123
80%	0.9811	0.6138
85%	0.9825	0.6216
90%	0.9833	0.6224
95%	0.9688	0.6248
100%	0.9296	0.6477

## 6. Conclusions

Currency rules can be used for both data repairing and data quality evaluation. Based on the current research on data currency, this paper further studied the dynamic updating and parallelization algorithms of currency rules, and proposed several data currency quality evaluation models. The main research results are as follows:

- 1 Extending the basic currency rule. The extended currency rules can be updated incrementally, and the new added path length attribute can provide

more effective information for currency quality evaluation.

- 2 Proposing parallel extraction and merging algorithms for currency rules. Experimental tests show that the parallel algorithms are effective. Compared with non-parallel algorithms, the extraction performance is improved by 75.20%, merging performance is improved by 55.23%, and the evaluation performance is improved by 85.86%.
- 3 Proposing the CR, SM, CMs data currency evaluation models. Experimental results show that the precisions of the four models are relatively high, and all of them can meet the needs of general data currency quality evaluation. The CM2 model which references both the path length and the support features of the rules has the best performance, when recall is 90%, precision is 0.9833, and when recall is 95% F1 score is 0.9695.

In future data currency research, we will explore data repairing and quality evaluation methods that combine conditional function dependence and data curren-

cy rules. The currency rules can be considered as some features extracted from the sequential relationship of the data; the conditional function dependence can be considered as some features extracted from the relationships between the data fields. The fusion research on data sequence relationship and data field relationship will help to further improve data repair and improve data quality evaluation effect. Research on the fusion of two feature aspects of the vertical sequence relationships and the horizontal fields relationships will help to further improve the quality of data repairing and data currency evaluation.

### Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant No. 61772352; National Key Research and Development Project under Grant No. 2020YFB1711800 and 2020YFB1707900; the Science and Technology Project of Sichuan Province under Grant No. 2019YFG0400, 2020YFG0479, 2020YFG0322, and the R&D Project of Chengdu City under Grant No. 2019-YF05-01790-GX.

## References

1. Brkić, L., Mekterović, I. A Time-Constrained Algorithm for Integration Testing in a Data Warehouse Environment. *Information Technology and Control*, 2018, 47(1), 5-25. <https://doi.org/10.5755/j01.itc.47.1.18171>
2. Ding, X., Wang, H., Gao, Y., Li, J., Gao, H. Determining the Currency of Dynamic Data. In: *Proceedings of the ACM Turing, Celebration Conference*, ACM Press, 2017, 17. <https://doi.org/10.1145/3063955.3063972>
3. Ding, X., Wang, H., Gao, Y., Li, J., Gao, H. Efficient Currency Determination Algorithms for Dynamic Data. *Tsinghua Science and Technology*, 2017, 22 (3), 227-242. <https://doi.org/10.23919/TST.2017.7914196>
4. Ding, X., Wang, H., Zhang, X., et al. Association Relationships Study of Multi-dimensional Data Quality. *Ruan Jian Xue Bao/Journal of Software*, 2016, 27(7), 1626-1644. <https://doi.org/10.13328/j.cnki.jos.005040>
5. Du, Y., Shen, D., Nie, T., et al. A cleaning method for consistency and currency in related data. *Chinese Journal of Computers*, 2017, 40 (1), 92-106. <https://doi.org/10.11897/SP.J.1016.2017.00092>
6. Duan, X., Guo, B., Shen, Y., et al. Data repair Algorithm Based on Currency Rules. *Ruan Jian Xue Bao/Journal of Software*, 2019, 30(3), 589-603. <https://doi.org/10.13328/j.cnki.jos.005688>
7. Eckerson, W. *Data Quality and the Bottom Line: Achieving Business Success Through a Commitment to High Quality Data*. Washington: The Data Warehouse Institute, 2002.
8. Fan, W., Geerts, F. *Foundations of Data Quality Management*. Morgan & Claypool Publishers, 2012.
9. Fan, W., Geerts, F., Wijzen, J. Determining the Currency of Data. *ACM Transactions on Database Systems*, 2012, 37(4), 1-46. <https://doi.org/10.1145/2389241.2389244>
10. Fan, W., Geerts, F., Wijzen, J. Determining the Currency of Data. In: *Proceedings of the 30th ACM Sigmod-sigact-sigart Symposium on Principles of Database Systems*. ACM Press, 2011, 71-82. <https://doi.org/10.1145/1989284.1989295>
11. Fan, W., Geerts, F., Yu, W. Conflict Resolution with Data Currency and Consistency. *Journal of Data & Information Quality*, 2014, 5(1-2), 6. <https://doi.org/10.1145/2631923>
12. Fuhr, N., Rölleke, T. A Probabilistic Relational Algebra for the Integration of Information Retrieval

- and Database Systems. *ACM Transactions on Information Systems*, 1997, 15(1), 32-66. <https://doi.org/10.1145/239041.239045>
13. Guo, B., Li, Q., Duan, X.-L., Shen, Y.-C., Dong, X.-Q., Zhang, H., Shen, Y., Zhang, Z.-L., Luo, J. Personal Data Bank: A New Mode of Personal Big Data Asset Management and Value-Added Services Based on Bank Architecture. *Chinese Journal of Computers*, 2017, 40(1), 126-143. <https://doi.org/10.11897/SP.J.1016.2017.00126>
  14. Huang, S.-Y., Huang, S.-M., Wu, T.-H., Hsieh, T.-Y. The Data Quality Evaluation of Graph Information. *Journal of Computer Information Systems*, 2011, 51(4), 81-91.
  15. Huo, R., Wang, H., Zhu, R., Li, J.-Z., Gao, H. Map-reduce Based Entity Identification in Big Data. *Journal of Computer Research and Development*, 2013, 50(s2), 170-179.
  16. Jin, C., Liu, H., Zhou, A. Functional Dependency and Conditional Constraint Based Data Repair. *Ruan Jian Xue Bao/Journal of Software*, 2016, 27(7), 1671-1684. <https://doi.org/10.13328/j.cnki.jos.005037>
  17. Koubarakis, M. Representation and Querying in Temporal Databases: The Power of Temporal Constraints. In: *Proceedings of the International Conference on Data Engineering*. IEEE Computer Society, 1993. 327-334. <https://doi.org/10.1109/ICDE.1993.344049>
  18. Li, M., Li, J. A Minimized-Rule Based Approach for Improving Data Currency. *Journal of Combinatorial Optimization*, 2016, 32(3), 812-841. <https://doi.org/10.1007/s10878-015-9904-8>
  19. Li, M., Li, J., Gao, H. Evaluation of data currency. *Chinese Journal of Computers*, 2012, 35 (11), 2348-2360. <https://doi.org/10.3724/SP.J.1016.2012.02348>
  20. Li, M., Li, J. Algorithms for Improving Data Currency. *Journal of Computer Research and Development*, 2015, 52(9), 1992-2001. <https://doi.org/10.7544/issn1000-1239.2015.20140687>
  21. Liang, Y., Duan, X., Ding, Y., Kou, X., Huang, J. Data Mining of Students' Course Selection Based on Currency Rules and Decision Tree. In *Proceedings of the 2019 4th International Conference on Big Data and Computing*. ACM, New York, NY, USA, 2019, 247-252. <https://doi.org/10.1145/3335484.3335541>
  22. Meyden, V. The Complexity of Querying Indefinite Data About Linearly Ordered Domains. *Journal of Computer & System Sciences*, 1997, 54(1), 113-135. <https://doi.org/10.1006/jcss.1997.1455>
  23. Sargent, P. Data Quality in Materials Information Systems. *Computer-Aided Design*, 1992, 24(9), 477-490. [https://doi.org/10.1016/0010-4485\(92\)90028-9](https://doi.org/10.1016/0010-4485(92)90028-9)
  24. Savulionienė, L., Sakalauskas, L. A Stochastic Algorithm of Frequent Set Search for Mining Association Rules. *Information Technology and Control*, 2014, 43(2), 121-132. <https://doi.org/10.5755/j01.itc.43.2.3135>
  25. Song, S., Cao, Y., Wang, J. Cleaning Timestamps with Temporal Constraints. *Proceedings of the VLDB Endowment*, 2016, 9(10), 708-719. <https://doi.org/10.14778/2977797.2977798>
  26. Wang, H., Fan, W. Object Identification on Complex Data: A Survey. *Chinese Journal of Computers*, 2011, 34(10), 1843-1852. <https://doi.org/10.3724/SP.J.1016.2011.01843>
  27. Wang, R., Strong, D. Beyond Accuracy: What Data Quality Means to Data Consumers. *Journal of Management Information Systems*, 1996, 12(4), 5-33. <https://doi.org/10.1080/07421222.1996.11518099>
  28. Zhang, H., Guo, B., Shen, Y.-C., Duan, X.-L., Dong, X.-Q. An Information Source Localization Algorithm Based on Cellular Automata Model. *International Journal of Modern Physics B*, 2019, 33(28), 1950336-1-14. <https://doi.org/10.1142/S0217979219503363>
  29. Zhang, H., Diao, Y., Immerman, N. Recognizing Patterns in Streams with Imprecise Timestamps. *Proceedings of the VLDB Endowment*, 2010, 3(1-2), 244-255. <https://doi.org/10.1016/j.is.2012.01.002>
  30. Zhou, A.-Y., Jin, C.-Q., Wang, G.-R, Li, J.-Z. A Survey on the Management of Uncertain Data. *Chinese Journal of Computers*, 2009, 32(1), 1-16. <https://doi.org/10.3724/SP.J.1016.2009.00001>

