DOI: 10.3745/KIPSTD.2010.17D.6.405

규칙기반 시스템에 사용되는 규칙 간소화 알고리즘

BaoWei Zheng[†] · 여 정 모[††]

요 약

다양한 업무요소들의 값의 조합에 따라 대상 값이 결정되는 것을 규칙이라고 한다. 업무를 표현한 기업의 정보시스템은 이러한 수많은 규칙들을 포함하는데, 이러한 규칙들을 구현한 서버 시스템을 규칙기반 시스템이라고 한다. 규칙기반 시스템은 규칙 엔진 기법을 사용하거나 직접 데이터베이스를 사용하여 구현된다. 규칙 엔진 기법은 많은 단점을 가지기 때문에 대부분 관계형 데이터베이스를 사용하여 규칙기반 시스템을 구현한다. 업무의 규모가 커지고 복잡하게 될수록 수많은 다양한 경우의 규칙이 존재하게 되므로 시간과 비용이 크게 증가하고, 대량의 저장공간을 요구하게 될 뿐만 아니라 수행속도의 저하 현상도 많이 발생한다. 따라서 본 연구에서는 이러한 수많은 경우의 규칙들을 동일한 효과를 가지는 간소화된 경우의 규칙들로 변환시킬 수 있는 알고리즘을 제안한다. 본 연구의 알고리즘을 가지고 다양한 업무 규칙 데이터에 적용하여 테스트한 결과 데이터 건수를 간소화시킬 수 있음을 입증하였다. 본 연구의 알고리즘을 사용하여 업무 규칙 데이터를 간소화하게 되면 데이터베이스를 사용하여 구현된 규칙기반 시스템의 성능을 개선할 수 있다.

키워드 : 업무요소, 규칙, 규칙기반 시스템, 관계형 데이터베이스, 규칙엔진

The Rule Case Simplification Algorithm to be used in a Rule-Based System

BaoWei Zheng[†] · Jeongmo Yeo[††]

ABSTRACT

A rule is defined as a case to determine the target values according to combination of various Business factors. The information system is used to represent enterprise's business, which includes and implements the amount of these rules to Rule-Based System. A Rule-Based System can be constructed by using the rules engine method or Relational Database technology. Because the rules engine method has some disadvantages, the Rule-Based System is mostly developed with Relational Database technology. When business scales become larger and more complex, a large number of various rule cases must be operated in system, and processing these rule cases requires additional time, overhead and storage space, and the speed of execution slows down. To solve these problems, we propose a simplification algorithm that converts a large amount of rule cases to simplification rule cases with same effects. The proposed algorithm is applied to hypothetical business rule data and a large number of simplification experiments and tests are conducted. The final results proved that the number of rows can be reduced to some extent. The proposed algorithm can be used to simplify business rule data for improving performance of the Rule-Based System implemented with the Relational Database.

Keywords : Business Factor, Rule, Rule-Based System, Relational Database, Rules Engine

1. Introduction

The Rule-Based System represents information in terms of a group of rules that tell us what we should do in different situations. A general Rule-Based System consists of a number of rules and facts and rules engine that is developed by procedural language. There are two kinds of Rule-Based System, a forward chaining system and a backward chaining system. In a forward chaining system, we start drawing target results with the initial facts, and keep using the rules to draw new conclusions based on given facts. In a backward chaining system, we start

† 준 회 원 : 부경대학교 정보공학과 박사수료
†† 정 회 원 : 부경대학교 컴퓨터공학과 교수
논문접수 : 2010년 6월 7일
수 정 일 : 1차 2010년 8월 16일
심사완료 : 2010년 9월 18일

drawing rules that include the hypotheses (or goal) designated by the users [2, 5].

There are many disadvantages in the Rule-Based System implemented with the rules engine method, mainly in three aspects. First, it needs complex rule patterns and rule reference mechanisms. Second, it requires much intuition and domain experience, and knowledge acquisition still constitutes a bottleneck in many potential applications. Third, it is difficult to change the structure of Rules engine as the business rules need to evolve.

Three major advantages of the Rule-Based System implemented with database are: first, data and the database can be directly used instead of complex rule patterns and rule reference mechanisms. Second, any kinds of rules can be represented by the data model. It means all the business rules can be processed in the system as long as it can be designed by the data model. Third, users need to merely modify SQL statement in order to satisfy the requirement for changing the business rules.

Because the amount of row cases is determined by the number of factors and domain degree, while the number of various rule cases is large, the system requires additional time, overhead and storage space for processing these rule cases, and the speed of execution also slows down. In order to solve the problem, we proposed a rule case simplification algorithm refer to a part of the Quine McCluskey tabulation method.

Since the new algorithm can be applied to reduce the amount of rule cases the efficiency of processing rule cases also can be advanced to a certain extent [3].

To simplify the complexity of the simplification algorithm, the algorithm is divided into three parts, and each part will be introduced sequentially. We will prove that when the algorithm is implemented on the basis of one different factor, the result of method that excludes the rows of flag=0 is the same as the result of the method that includes the rows of flag=0 in middle of the executed table.

2. Related Work

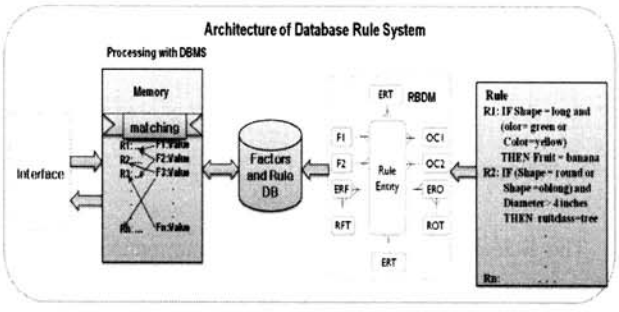
2.1 Rule-Based System Implemented with Database

In the Rule-Based System implemented with database, it is not necessary to develop a rules engine because the database is used instead of rules engine for processing all the rules, which had been executed by rules engine previously. A factor is regarded as a column in the rule case table, and the values of the factors are combined to make a rule case. And every rule case is considered as a row to save into rule case table. Therefore, the database is used not only for saving rules but also for completing the task of rules engine.

The specific architecture of Rule-Based System implemented with database is described in (Fig. 1). In the system, all the rule elements can be designed into a rule base data model according to the technology of Relational Database. As shown in (Fig. 1), these data models are named RBDM. And every entity in the rule base data model can be converted into a physical table that is saved into the database. The user can input certain factor values through the interface of the system. After some factors values are inputted into memory, DBMS first starts checking to find the rules whose factor value is equal to inputted values, if they are not found in memory, it accesses disk. In fact this operation is a procedure of using inputted factor values to find rules and target values; it is called matching operation. [4, 8].

The Rule-Based System implemented with database accommodates a great number of possible outcomes based upon various combinations of factors. But it does not accommodate a small number of possible outcomes based upon various combinations of factors because it is more efficient not to use the database.

However, because the matching operation is executed in memory, when the rules to be matched have many factors and each Factor has many values, the matching operation will need a very large memory space and the matching speed will becomes very slow. For example, there is a rule which has 5 factors and domain degree of each factor such as 100, 150, 200, 40, 170; the amount of data is



NOTE: F1, F2→Factor, ERF →Entity Rule Factor, ERT →Entity Rule Type, RFT →Rule Factor Type, OC1, OC2 → Outcome, ERO →Entity RULE Outcome, ROT →Rule Outcome type, RDB → Rule Database

(Fig. 1) The architecture of Database Rule System.

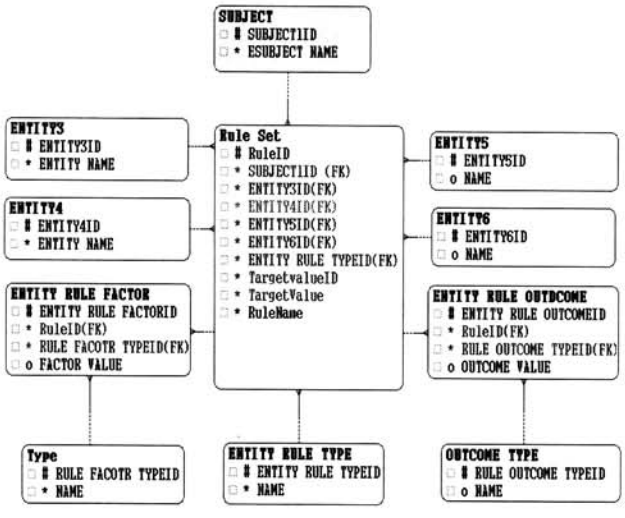
100*150*200*40*170=20,400,000,000 rows. This is only a very simple example; however, practical rules have many Factors and factors have big domain degree, therefore, the amount of data will be so large that the speed of execution is severely affected.

2.2 Pattern of Rule Data Model

According to the characteristic of rules, the pattern of rule data model can be classified into three types: Multi Dimension Data Model, Rule Matrix Data Model, and Rule Tree Data Model. Here we will examine the characteristics, adaptive situations, advantages and weaknesses of pattern of rule data model in detail. Because the Multi Dimension and the Rule Matrix are the most popular patterns for specifying general rules in practical application. Thus, the simplification algorithm can be applied in the two situations [6].

The Multi Dimension model is shown in (Fig. 2); the Multi Dimension Data Model maintains business rules data such as the business rule statement, business rule factors, and business rule outcomes for particular situation, problem area, or business subject area. Many situations such as this one occur during data modeling, and this pattern allows data professionals to address each situation with a common template to model rules [12, 13].

However, the Multi Dimension Data Model has the following weakness: if a new Factor is discovered, it is often easy for a data professional just to add that Factor as rule factor type without trying to find out if the enterprise data model captures this factor as an entity in its own right. There may need to be a more flexible way

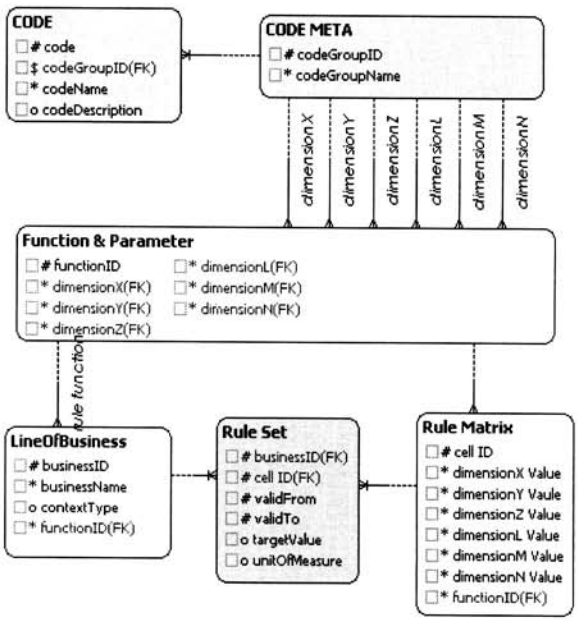


(Fig. 2) Multi Dimension Data Model.

to classify the rules. There are many types of data that are common to various types of business rules and thus could be consolidated into a business rule super-type [7, 8].

(Fig. 3) shows the Rule Matrix Data Model. All the factors that affect the target value of rule cases are categorized into different groups, using an entity named CODE META to manage them. All the factors are managed by CODE entity. Factors affecting the target value can be considered variables which can be inherited from CODE META entity. The function of determining target value and necessary variables is considered a row to manage in Function & Parameter entity. Every specific rule case as a row can be saved in the Rule Matrix.

The largest advantage of the Rule Matrix Data Model is that it can be applied in any situations. However, the largest disadvantage of the Rule Matrix Data Model is that it cannot be considered a flexible part inserted into the existing model. It is difficult to add or delete factors to or from the Rule Matrix Data Model according to the changes of business rules [1, 10].



(Fig. 3) Rule Matrix Data Model.

2.3 Redundancy

Redundant rule cases may be included in rule cases, these redundant rule cases have no influence on correctness and functionality of Rule-Based System, but they can slow down execution speed and occupy memory. A general technique for dealing with redundancy is to detect and remove redundant rule cases in previous

studies. But it is difficult to use the method of identifying identical rule cases, and subsumed rule cases, and equivalent rule cases, and unusable rule cases for detecting redundant rule cases.

3. Rule Case Simplification Algorithm

This section introduces principle design, and related variables, and definition, and detailed procedure of the simplification algorithm.

3.1 Principle Design of Simplification Algorithm

The simplification algorithm is defined as a procedure that simplifies redundant rule cases, merges the repeated values of Factors and reduces the amount of rule cases.

We regard a prime implicanct as a row of rule case table. To find all the prime implicants, compare each minterm in the canonical form with all its other minterms in the tabulation method. And to find all simplified rule cases, we compare all rule cases that differ in the value of only one factor in the simplification algorithm. Variables of tabulation method are replaced with factors of rule. Grouping the same number of variables whose values are equal to 1 in the tabulation method, every row of rule case table will be considered a group in simplification algorithm.

To reduce the amount of compared times, all the rows of the rule case table are classified into different groups according to target value and the simplification algorithm is executed for every group table. Because the tabulation method is used for minterms of Boolean functions, variable can only take 0 or 1, and there are only two minterms that differ in the value of only one variable. However, because the factor of rule can take many values, the value of different factor is replaced with an underscore only when the different factor takes the whole values in the simplification algorithm. This process is repeated until no new subcubes can be generated or until all the minterms have been grouped into n-subcube in the tabulation method. The simplification algorithm will be executed repeatedly until all no rule cases can be simplified in the simplification algorithm [16].

3.2 Definition of Related Variables

Factor1, factor2, ⋯, factorm are various factors affecting the target value, and m is the number of factors. When comparing any two rows, a factor which takes different values in two compared rows is called "different factor", other factors which take the same values in two compared rows are called "same factor". I is a loop variable which we call "reference index". It indicates the reference row of the rule case table. Domain of i is {i| i=1⋯n}. J also is a loop variable named "different index". J= (i+1⋯n) and when there is only one different factor in two compared rows, j has the meaning of its existence. It indicates a different row of the rule case table.

Simplification algorithm will use different index to compare with reference row. K also is a loop variable named "compared index". K= (j+1⋯n) and if there is only one different factor in i and j, then k starts looping. K indicates compared row2 in the rule case table. A is a two-dimension array. We use it to save different values of the different factor and it's RuleID. Flag is defined as an additional column of Rule case table, it can take 1 or 0, if a row is used by the simplified procedure, then give 1 to flag, otherwise, give 0 to flag.

Group table is a data set that classify original rule case table according to target value. The simplified table is a result set obtained after implementing the simplification algorithm. If a group table has been simplified completely, and then the result of simplifying group table is a simplified table. Note that we have to create a table in the identical structure with the original rule case table. The same refers to the type of table, column, and column domain, etc. [11, 15].

3.3 Definition of Simplification Algorithm

Definition1. The rule R is logically redundant, if there exists rule R″ obtained from rule R by simplifying some rule cases rc and there is R|=R″ and R″|=R.

In other words, R is redundant if there is a possibility to reduce the number of rule cases by simplifying redundant rule cases, and R″ obtained in this way is logically equivalent to the initial R.

Definition2. $R=r_1 \wedge r_2 \cdots \wedge r_n$, $R' = r_1 \wedge r_2 \cdots \wedge r_m$, and $R' \subset R$. Where $r_i=\phi_i \rightarrow h_i$ are some rule cases for i=1, 2, ⋯, n. Precondition of R is ϕ, $\phi=\{ f_1, f_2, \cdots, f_m\}$, and h_i are target values of the rules. Let there be a factor in the precondition ϕ, say f_i and $f_i=\{a, b, \cdots, p\}$. Let R′ have a different factor f_i', and $f_i'=\{x, y, \cdots, q\}$. If and only if $f_i'=f_i$, R′ can be simplified to R″.

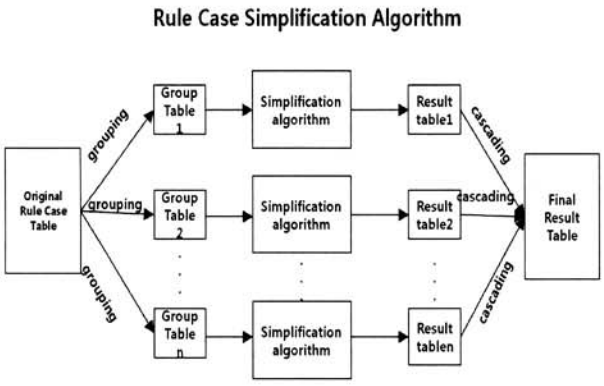
Obviously R″ is identical to R′. Further, m-1 rule

cases can be removed from R′ so that R″ is obtained and R″ ≡ R′.

3.4 Rule Case Simplification Algorithm

The procedure of the whole algorithm refers to three big operations that are shown in (Fig. 4).

1. Classifying all the rows of the original rule case table into different groups according to target value.
2. Executing simplification algorithm for each group table and generate a result table.
3. Cascading the result table that has been generated above and generate a final result table [9].

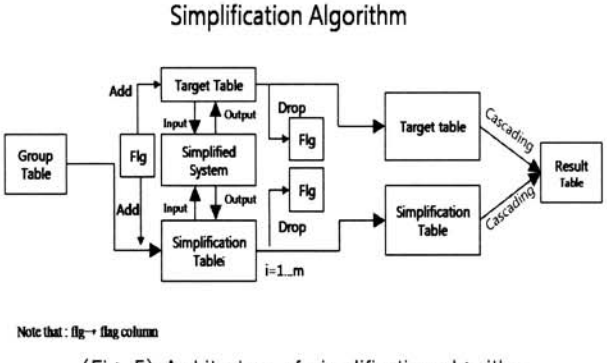


(Fig. 4) Architecture of rule case Simplification algorithm

3.5 Simplification Algorithm

The procedure of the whole algorithm is implemented through five operations that are shown in (Fig. 5).

1. Creating a target table in the identical structure with the group table, initialize it and obtain the object simplification table. The group table can be considered as the initialized simplification table.
2. Adding the flag column to the target table and simplification table
3. Executing the simplification system and generating the final target table; the final simplification table includes flag column.
4. Dropping the flag column from the final target table and the final simplification table, and generate the final target table and the final simplification table.
5. Cascading the final target table and the final simplification table into the final result table.

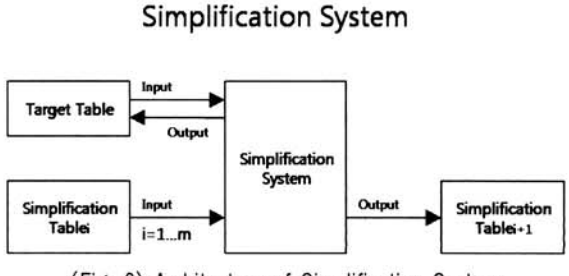


(Fig. 5) Architecture of simplification algorithm

3.6 Simplification System

Target table and un-simplification table provide rule cases to the simplification system, and the target table and simplification table save executed results of the simplification system. (Fig. 6) shows their procedure.

1. Giving all factor values of the first row of group table1 to the reference row.
2. If the reference and different rows satisfy the condition that there is one different factor in two rows, then give all factor values of i+1row of group table1 to the different row, comparing the values of reference row with the values of different row.
3. Only if there is one and same different factor in two compared rows, then save the two different values of the different factor into the first dimension of A array, and the RuleID values of reference row and different row are also saved into the two dimension of A array at the same time.
4. If the compared row2 such as reference row and different row has one and same different factor whose value is different in the compared rows and the values of other factors are the same, then start scanning all the rows from j+1in group <Table 1>.
5. If there is scanned value in A array, then scan the next row continuously. If there is no scanned value in A array, add it into the first dimension of A array. K's scan operation does not stop until it reaches the following two situations: the first, the number of values saved in the A array is equal to the number of the domain degree of the different



(Fig. 6) Architecture of Simplification System

factor. Second, all the rows have been scanned. This means that loop variable k has arrived the final row of group <Table 1> [15].

6. Create a simplification tablei in the structure identical to group table1, taking out values of the same factors from the reference row except for the different factor, replacing the values of the first dimension of an array with a "-" or null value. Insert the first value of f in the second dimension of A array into RuleID column, the same factor values out from the reference row into the corresponding column, and insert "-" or null into the different factor in simplification tablei.
7. Cascade repeated result tablei (i=1…n) of every group tablei(i=1…n) into the final result table[9].

3.7 Validation of Inferred Methods

Because the simplification algorithm compares any two rows based on only one different factor at the same time, in order to prove the validity of the method, we demonstrate it with the following example: for it, f1={y}, f2={a, b, c}, f3={m, n}, f4={z}. Rule case is represented in (Fig. 7).

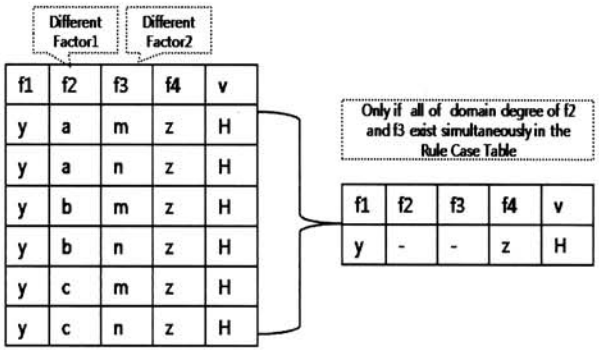
(Fig. 7) describes the procedure of simplification algorithm based on two different factors, in this situation only if domain degree of Factor f2 and Factor f3 exist simultaneously six rows, then the six rows can be simplified into a row. If any row among the six rows does not exist in the rule case table, then the simplified operation cannot be executed. In fact, all combinations of two different factor values occur seldom in practice. If we design the simplification algorithm based on two factors, the amount of rows that can be simplified will be reduced to a great degree and the efficiency of the algorithm also will be reduced largely. However, whether the algorithm based on one different factor, two or more Factors the final result is the same.

(Fig. 8) describes the procedure of simplification algorithm based on one different factor. Only if domain degree of Factor f3 exists simultaneously, the two rows can be simplified into a row in the first step, and then the simplified three rows can be simplified into one row in the second step. In this situation only two rows exist simultaneously to satisfy the conditions of the Simplification Algorithm. In fact, there is a larger probability of existing simultaneously all the combinations of one different factor values in practice. Thus we have designed the simplification algorithm based on one factor.

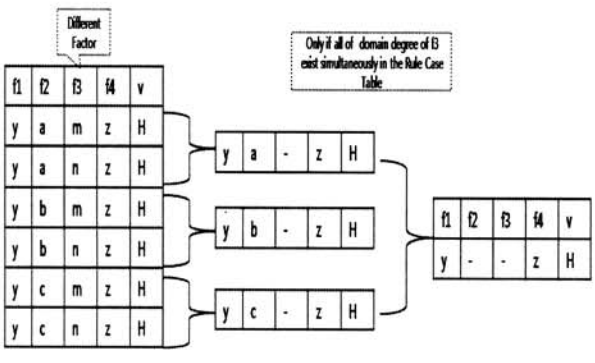
This method not only advances the efficiency of the simplification algorithm but also increases the amount of rows that can be simplified. Whether the algorithm is based on one Factor, two or more factors the final result is the same.

In order to avoid executing the inefficient way of comparing and reducing the amount of rows through a large number of simplifying experiments and tests, we found out that if the rows of the flag column value are marked as 0 in the first simplified step, then these rows neither can be simplified in the first simplified step nor can be simplified in all the simplified steps. Thus these rows are taken out directly from the every simplification table and insert them into target table when finishing every step of simplifying. This method improved the efficiency of the simplification algorithm largely.

However, to prove the validity of the method, we continue to use the previous example for proving the correctness and reasonableness of the method of taking out all rows of the flag column whose values have been marked as 0 and inserted directly into target table, as well as the method of including all rows of the flag



(Fig. 7) Procedure of two different factors



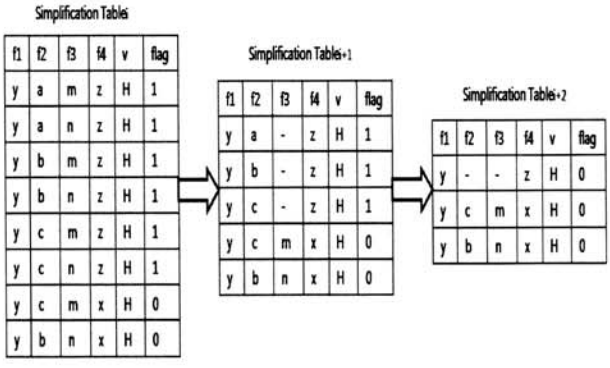
(Fig. 8) Procedure of one different factor

column whose values have been marked as 0 in every simplified step.

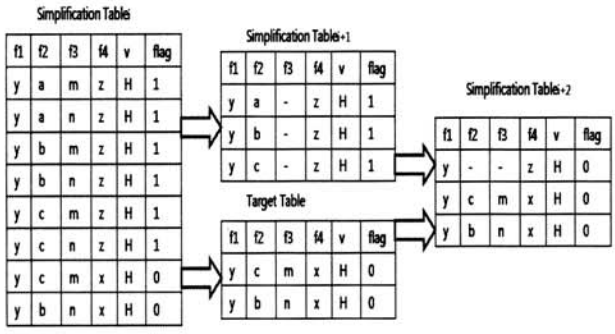
When finishing the first loop of simplifying operation, rows that have already been simplified are marked by 1and rows that cannot be simplified are marked by 0 in flag column, as in the flag values of simplification tablei in (Fig. 9). As shown in (Fig. 9), the rows whose flag column value is 1 in simplification tablei has been already simplified in the first executed step, rows whose flag column value is 0 have not been simplified in the first executed step. Taking out these rows and inserting them directly into simplification tablei+1, these rows are compared in the next simplified step.

Disadvantages of the method are referring to increase the amount of rows that do not have to be executed and affecting severely efficiency of the simplification algorithm. The final table of the example is the simplification tablei+2 which has three rows, as show in (Fig. 9).

The method of including target table is similar to the method of excluding target table. The difference between two methods is that the simplification algorithm takes out rows whose flag column value is 1 from simplification tablei and inserts them into simplification tablei+1, and it takes out rows whose flag column value is 0 from simplification tablei and inserts them into target table. This means that rows of target table do not have to be simplified by simplification algorithm from this step, but rows of simplification tablei+1 only have to be simplified in next step, as show in (Fig. 10). This method can reduce the amount of rows of simplifying significantly. Thus the method including target table has more efficient efficiency than the method excluding target table in a situation of returning the same result table.



(Fig. 9) Method excluding Target Table



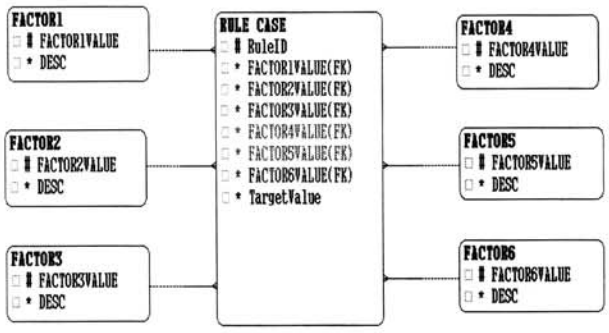
(Fig. 10) Method including target table

4. Results

To make a simplification algorithm that can be applied to more generalized type of rule, we make use of a generalized data model of rule in this paper and needless items have been already deleted from the Multi Dimension Data Model. The structure of generalized data model of rule is described in (Fig. 11).

There are 6 factor entities and 1 rule case entity in (Fig. 11), each of the factors will be converted into corresponding table, and rule case entity will be converted into rule case table. Because rule case entity inherits 6 foreign keys from 6 factor entities, every foreign key will be converted into a different column that belongs to the rule case table, and every foreign key represents a factor that affects the end result of rule.

In (Fig. 11), the column of the rule case table consists of column BuleID, 6 foreign keys and TargetValue; RuleID is the identifier of a rule case and it uniquely identifies each rule case in the table. Rows of the rule case table are possible combination of all the factors. A factor table for saving factor values without allowing any repeat rows in it. Domain degree indicates the number of values in the domain of $factor_i$ {value| value=1…n}. TargetValue column



(Fig. 11) Generalized data model of rule

includes many goal values, which are determined by factor values, and it is abbreviated as V.

In order to test the effect of the simplification algorithm, we developed an application program with C++ language and oracle DBMS. The application program can access the practical rule case table from oracle database and save it into memory. Then the rule case table is considered as an object table for executing the simplification algorithm. Finally, the simplified rule cases are used into Rule-Based System.

Through a large number of simplifying experiments and tests, we found that simplified degree varies with the type of table. And the most important element that determines the type of table is the business rule. Because we cannot change the business rule for advancing the efficiency of the simplification algorithm, the effect of the algorithm varies with the type of business rule. The simplified effect of the simplification algorithm for various business rules is listed in <Table 1>.

The results of the simplification algorithm can be classified into two parts according to its characteristics. The first part refers to the number of rows that vary with the number of factors in rule case table. We take a bank charges rule case here as an example. Before and after simplification, the specific changes in the number of rows are shown in (Fig. 12). The greater the number of factors becomes, the less number of rows can be simplified. Distance between the before simplification line and after simplification line becomes larger.

Because the number of rows also varies with the number of domain degree of factors, let domain degree of some factors (marked as *) be increased in random order. Between the before and after simplification, the specific changes in the simplified effect are shown in (Fig. 13).

The number of simplifying rows decreases with increasing number of domain degrees.

The simplification ratio achieved in a given environment depends on the nature of the rules and the domain degree of every factor.

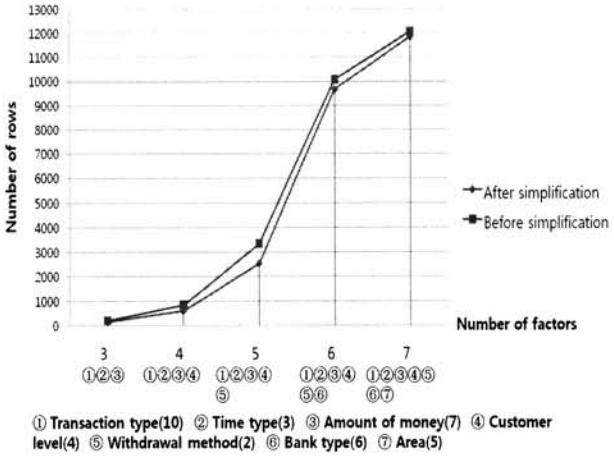
To evaluate efficiency of the simplification algorithm, we have conducted simplification experiments separately in the database and in the file system and results are shown in (Fig. 14). The file system takes less time in simplifying the same number of rule cases when the amount of rule cases is small; but the database takes less time in simplifying the same number of rule cases when the amount of rule cases is large.

Because the simplified rule cases obtained from the simplification algorithm can be permanently used in Rule-Based System when the rules cannot be changed, the simplification algorithm just needs to be executed only once. In fact there is no sense in comparing the performance of database and file system.

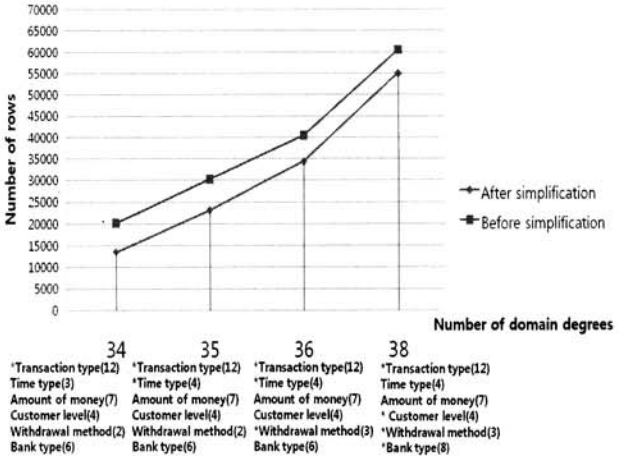
The simplification algorithm is mainly used for

〈Table 1〉 Simplified rate of various business rules

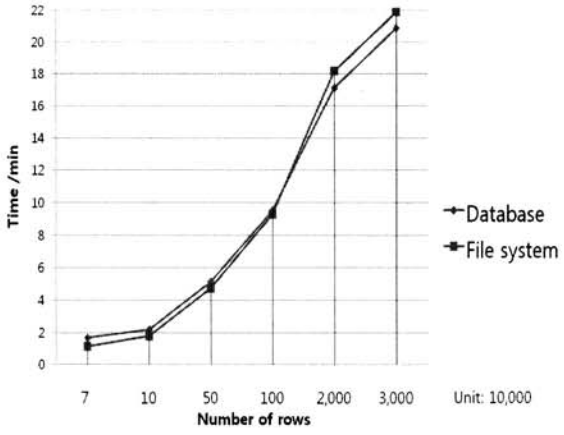
Business Rule type	Rows of Un-Simplification	Rows of Simplification
Bank charges rule	70	48
Insured cost rule	100	87
Law rule	130	116
Product rule	150	122
Order rule	200	174
Shipment rule	300	277



(Fig. 12) Number of factors impacts on simplified effect



(Fig. 13) Domain degree impacts on simplified effect



(Fig. 14) Time cost of simplification system in database and file system

simplifying the number of rule cases, so the simplified rule cases can be used in Rule-Based System implemented with database, also can be used in Rule-Based System implemented with rules engine. Because Rule-Based System processes a relatively fewer rule cases than before, the performance of Rule-Based System is improved a little.

5. Conclusion

In this paper we introduced the architecture of Rule-Based System implement with database and two types of rule data model, which can be applied to represent rules in suitable situations. In order to reduce the amount of rule cases that must be processing in Rule-Based System, we proposed the simplification algorithm. And the formal definition of the simplification algorithm also is made in this paper, and proved the validity of all the methods of referring to in the algorithm. A large number of simplifying experiments and tests are conducted; the final results proved that the number of rule cases can be reduced by the algorithm.

Although the simplification algorithm can reduce the number of rule cases a little, it can be used in specific situations rather than all situations. Thus, we still need to extend the application range of the simplification algorithm to all possible situations, and further research is called for to minimize the limitation of the simplification algorithm.

References

[1] En-core consulting, "Rule Base Data Model", 4th Seminar material, Apr. 2008.

[2] Janusz Kacprzyk, "Logical Functions for Rule-Base Systems, second edition", Spinger, 2006.

[3] Ahmed T. Sadik, "Premises Reduction of Rule Based Expert System Using Association Rules Technique", International Journal of Soft Computing, Vol.33, No.1, pp.195-200, 2008.

[4] Oracle Corp, "Oracle® Fusion Middleware User's Guide for Oracle Business Rules 11g Release 1 (11.1.1)", Oracle White Paper, pp.1-56, 2009.

[5] Alison Cawsey, "The architecture of forward chaining Rule-Base System and backward chaining Rule-Base System", Computing and Electrical Engineering Journal, Vol.14. No.3, pp.123-156, Dec., 2007.

[6] David C.Hay, "Data Model Patterns A Metadata Map", Morgan Kaufmann Publishing, pp.273-338, 2007.

[7] Len Silverston & Paul Agnew, "The Data Model Resource Book", Wiley Publishing, pp. 411-468, 2009.

[8] Steve Hoberman, "Data Modeling Master Class", Steve Hoberman & Associates LLC, pp. 112-280, 2008.

[9] Stephande Faroult & Peter Robson, "The Art of SQL", publishing House of electronics industry, pp.167-190, 2008.

[10] Lee Huw Sick, "New Written, Large Scale Database Solution", Publishing En-core consulting, pp.1-636, 2005.

[11] Malcolm Chisholm, "From How to Build a Business Rules Engine: Extending Application Functionality through Metadata Engineering", Morgan Kaufman Publishing, pp.1-50, 2004.

[12] L.Silverston, "The Model Resource Book Volumen1", Wiley Publishing, pp.133-180, 2001.

[13] Graeme Simsion, "Data Modeling Theory and Practice", Technics Publications LLC, pp.230-276, 2007.

[14] Kerry Patterson & Joseph Grenny, "Crucial Conversations: Tools for Talking When Stakes are High", McGraw-Hill Publishing, pp.125-210, 2002.

[15] InSung Kang, "Tree-Based Index Overlay in Hybrid Peer-to-Peer Systems", Journal of Computer Science and Technology, Vol.25, No.2, pp.179-180, 2009.

[16] P.W. Chandana Prasad, Azam Beg, Ashutosh Kumar Singh, "Effect of Quine-McCluskey Simplification on Boolean Space Complexity", Conference on Innovative Technologies in Intelligent Systems and Industrial Applications, July, 2009.

BaoWei Zheng

e-mail : buging2007@hotmail.com

2004년 서안전자과학기술대학교 컴퓨터공학과(학사)

2007년 위덕대학교 컴퓨터멀티미디어공학과(공학석사)

2007년~현 재 부경대학교 정보공학과 (박사)

관심분야 : 데이터베이스, Rule-Based System, 모발일 인터페이스

여 정 모

e-mail : yeo@pknu.ac.kr

1980년 동아대학교 전자공학과(학사)

1982년 부산대학교 전자공학과(공학석사)

1993년 울산대학교 대학원 전자 및 전산기공학과(공학박사)

1986~현 재 부경대학교 컴퓨터공학과 교수

관심분야 : 데이터베이스, ITA/EA, 전자상거래, EMFG