Automated Data Warehousing for Rule-based CRM Systems

Han-joon Kim  TaeHee Lee  Sang-goo Lee  Jonghun Chun

School of Computer Science and Engineering
Seoul National University,
San 56-1 Shillim-Dong Kwanak-ku, Seoul, 151-742 Korea,
Email: hjkim@europa.snu.ac.kr

Abstract

This paper proposes a novel way of automatically developing data warehouse configuration in rule-based CRM systems. Rule-based CRM systems assume that marketing activities are represented as a set of IF-THEN rules. Currently, to provide good quality CRM functionalities, CRM systems seek to combine conventional CRM methodologies with data warehousing technology. A data warehouse can be abstractly seen as a set of materialized views. Selecting views for materialization in a data warehouse is one of the important decision-making tasks in its design. However, there are few facilities in CRM systems with respect to data warehouse design that alleviate the problems associated with data schema maintenance. Given a set of campaign rules expressing marketing strategies, the proposed method generates data warehouse configuration (including database schema and indexing constraint) that can satisfy all the input rules. Our method begins on the premise that data warehouse configuration can be reversibly extracted from marketing campaign rules. This method includes algorithms for database schema generation, indexing constraint generation, schema normalization for removing data redundancies, and OLAP (On-Line Analytic Processing) query generation.

Keywords: Data warehouse, CRM, Rules, Materialized view, Analysis query, Star-join index

1 Introduction

In today’s highly competitive business environment, CRM (Customer Relationship Management) systems, which provide the framework for analyzing customer profitability and improving marketing effectiveness, have become an indispensable component in enterprise information systems. Typically, CRM activities include data analysis, campaign design, response analysis of customer data. To effectively support such activities, a data warehouse (which is a repository that integrates information from multiple operational data sources) must be developed to act as the backbone of CRM systems. A data warehouse is a core part that determines the performance of CRM systems and quality of CRM services (McKnight 2001). A data warehouse for CRM should be customer-centric: that is, it should provide a unified view of customer data such as customer demographic data, customer channel data, campaign data, and response data.

Unfortunately, many data warehousing projects have led to failure without having to reap tangible results. One of the reasons for such failure is that the data warehouse systems developed nowadays are unable to adapt well to the rapidly changing operational environment. As in developing other software systems, building data warehouse systems requires requirement analysis on ‘what the system is used for?’ or ‘what information should be managed?’. However, in performing data warehousing, it is very difficult to extract sufficient requirements from users before the system is used. We often do not precisely know what the problems actually are that we are trying to solve. Data warehouse systems are unlike other operational systems in that it is not possible to define the requirements precisely. It is only after the system is used that users realize its practicality, and then recognize what is lacking. And, the new requirements lead to the update of the current data warehouse schema.

A data warehouse stores materialized views to provide fast and uniform access to distributed operational data sources. Selecting views for materialization in a data warehouse is one of the important decision-making tasks in its design. Building a data warehouse is a very complex task requiring a variety of different skills to carry out such a task. One of the most difficult ones is to keep it up-to-date with respect to the rapidly changing operational conditions of usage environment (O’Gorma et al. 1999). Therefore, a conventional strategy for data warehouse development is that we consider all possible information that could be used in the future. Unfortunately, this strategy results in significant space overhead and low space utilization. Furthermore, although we try to gather all possible information, we still have to update the data warehouse configuration whenever new requirements are introduced. Such a task is highly costly and time-consuming.

In data warehouse design, there are few facilities in CRM systems that alleviate the problems associated with data schema maintenance. To overcome this problem, we propose a method that helps automatically generate data warehouse configuration under rule-based CRM system environment. The proposed method is motivated by the fact that the usage pattern of data warehouses is implicitly expressed in CRM activities such as campaign design and campaign analysis. The act of performing a campaign for a specified group of customers imply that the marketer involved with the campaign knows what kind of information should be extracted from operational data sources for executing the campaign and for analyzing its results. In rule-based CRM systems, a campaign (or marketing strategy) can be represented as a set of IF-THEN rules. Since campaign rules include the conditions for some target customers, they become important factors for analyzing the effectiveness of campaigns performed. This means data warehouse configuration can be reversibly extracted from marketing campaign rules. Only by designing campaign rules, all schemas of materialized views can be automatically generated and can evolve while satisfi-
ing the current requirements. Our proposed method includes automated generation of materialized view schema, indexing constraint scheme, and predefined analysis queries.

The remainder of this paper is organized as follows. In Section 2, the problem that must be overcome is briefly introduced. In Section 3, we provide an overview of rule-based CRM system and its related definitions. In Section 4, we present the algorithm of automated data warehousing. And, in Section 5, an example of automatically generated schemas will be presented, followed by reviews of related work, and finally, in Section 6, conclusions will be given.

2 Problem Statements

In this paper, we attempt to automatically develop data warehouse configuration in rule-based CRM systems. Given a set of campaign rules, we should be able to produce data warehouse configuration satisfying those rules. Data warehouse configuration consists of data warehouse schema, indexing constraint, and predefined analysis (or OLAP[^1]) queries.

In general, data warehouse schemas are developed according to the multi-dimensional modelling strategy, which have so-called ‘star schema’ generated. Star schema is composed of two types of tables: fact and dimension tables. Fact tables store the target information for analysis, and dimension tables store criteria (dimension) information for multi-dimensional analysis. For example, if we assume that we must analyze the sales amount of a certain shopping mall, we try to analyze the sales amount from various viewpoints of products, stores, time, and customers. The data warehouse schema that reflects such a requirement carries a fact table in which each record contains details such as the sales amount of a certain product purchased by a certain customer in a certain store at a certain time. It also has a dimension table corresponding to each of the analysis criteria mentioned above (i.e., products, stores, customers and time).

Once data warehouse schema is generated, indexing constraint and analysis queries can be easily extracted. Most of the queries against data warehouses accompany join operations between dimension tables and fact tables, and group-by operations by the values of fields (or attributes) occurred in dimension tables. Here, the cost incurred by the join operation may be very large since fact tables are usually of enormous size. Thus, to improve the performance of join operations, a special index called ‘star-join index’ between dimension tables and fact tables is prepared, which is a result of Cartesian product operation between related tables. In addition, a B-tree index for each field that participates in join operations needs to be generated to ensure efficient group-by operations. As for analysis queries, it is not easy to predict analysis queries that users need. However, if we build only the necessary data warehouse schema, useful analysis queries corresponding to each rule can be easily produced from the schema.

3 Rule-based CRM systems

Figure 1 presents the overall architecture of a rule-based CRM system, which is composed of three layers as follows.

- **CRM service layer** performs actual CRM services such as cross selling, personalized homepages, and direct Emailing.

- **Business intelligence layer** enables users to generate knowledge for CRM service through data mining and OLAP operation. Not only the data pattern knowledge but campaign rules defined based on the pattern knowledge are stored in rulebases. The rulebase manager constructs rulebases, and maintains them while performing OLAP analysis of the result of campaigns expressed in rules.

- **Integrated customer-centric database layer** is composed of three modules: update integrator, vocabulary manager, and schema manager. The update integrator integrates customer-relevant data from diverse operational data sources. The vocabulary manager helps users define basic terms used in defining campaign rules and generates a database schema for a view table corresponding to each term. The schema manager constructs and maintains data warehouses for OLAP analysis and data mining.

In this paper, we shall focus mainly on the Integrated customer-centric database layer.

![Figure 1: Architecture of rule-based CRM system](image1)

![Figure 2: Internal architecture of Integrated customer-centric database layer](image2)

We propose an internal architecture for the Integrated customer-centric database layer of rule-based CRM system, which manages multiple data sources, view tables and rulebases, as illustrated in Figure 2. In our work, vocabulary manager generates an SQL view table over operational data sources by interpreting each of the predefined terms. The data warehouse stores rule-aware star schema tables generated from given campaign rules by the schema manager (which will be explained later). Updates in data sources are propagated as update messages to the data warehouse.
and are integrated into view tables and a data warehouse in order to keep it up-to-date using the update integrator (or ETL\(^2\)) module.

In rule-based CRM systems, campaigns are represented in terms of **IF-THEN** rules, which are managed within rulebases. **IF-clause** describes a condition for isolating a group of customers, which is the target of a certain campaign whereas **THEN-clause** represents an actual campaign action to be executed. These two types of clauses are created with several pre-defined rule terms.

**Definition 1 (Rule terms)** A rule term \( v \) consists of subject, link, and semantics, which is denoted as \( v = (\text{subject}_v, \text{link}_v, \text{semantics}_v) \). The \( \text{subject}_v \) denotes the doer (or subject) of the action of generating instances of term \( v \), which is represented as a foreign key referencing instances corresponding to a subject of the term. The \( \text{link}_v \), is a foreign key referencing instances pointing to the value of the term. If the value of term \( v \) is a constant, \( \text{link}_v \), becomes null. The \( \text{semantics}_v \), represents the semantics of the term \( v \), which is represented as a combination of SQL statements and data manipulation function.

In Definition 1, data manipulation function is needed to describe semantics beyond the expressive power of SQL statements. The syntax for describing this function is determined by developing programming languages (e.g., Java and C++). For example, in the Java language, the value of 'age groups' can be computed by performing modular (%) operation with the value of 'age' returned by the SQL statement, “**SELECT** \( \text{CUSTOMER\_ID}, \text{AGE} \) **FROM** \( \text{CUSTOMER} \)”.

**Example 1** The term **Purchase\_items\_of\_last\_month** (which means a set of items that was purchased in the last month) is defined as follows.

\[
\text{PURCHASE\_ITEMS\_OF\_LAST\_MONTH} \equiv \\
\text{(C\_CUSTOMER\_ID, S\_PRODUCT\_ID,} \\
\text{SELECT} \\
\text{C\_CUSTOMER\_ID,} \\
\text{S\_PRODUCT\_ID} \\
\text{FROM} \text{CUSTOMER C, SALES S} \\
\text{WHERE} \\
\text{C\_CUSTOMER\_ID =} \\
\text{S\_CUSTOMER\_ID} \text{ AND} \\
\text{S\_PURCHASE\_YEAR = EXTRACT(YEAR FROM CURRENT\_DATE)} \text{ AND} \\
\text{PURCHASE\_MONTH = EXTRACT(MONTH FROM CURRENT\_DATE)-1} \\
\text{GROUP BY C\_CUSTOMER\_ID)}
\]

The subject of the rule term in Example 1 is “customer\_id” which denotes the doer of purchasing a certain product and its link is “product\_id” which represents objects of his or her purchase action. The function CURRENT\_DATE is one that returns date information (consisting of year, month, day, and time information) of the current day, and the function YEAR (or MONTH) is one that extract the year (or month) part from the given date information, which is specified in the standard SQL.

According to Definition 1, a rule term refers to a method of extracting immediately necessary information from operational data sources, and also expresses how it is associated with other vocabularies within a rule when interpreting a given rule. Rule terms specify the information of targets (or criteria) of campaign analysis, and they reflect the current usage of data warehouse and user requirements. The rule terms corresponding to targets and criteria of campaign analysis contribute to composing fact tables and dimension tables, respectively. Hereinafter, terms shall mean rule terms.

**Definition 2 (Campaign actions)** An action \( A \) is defined as \( (\text{method}_A, \text{args}_A, \text{measures}_A) \), where \( \text{method}_A \) is the method that performs a campaign action, \( \text{args}_A \) is a list of arguments necessary to compose the contents describing the campaign, and \( \text{measures}_A \) is a list of the rule terms that is the target of analysis after executing the campaign.

**Example 2** When a marketer attempts to recommend new products through Email, the campaign action named Email\_service can be defined as follows.

\[
\text{EMAIL\_SERVICE} \equiv \{\text{"Email\_sending"}, \{\text{"CUSTOMER\_NAME"}, \text{"PRODUCT\_ID"}, \{\text{"READ\_TIME"}, \text{"READ\_FREQUENCY"}, \text{"IS\_REJECT"\}}\}
\]

In Example 2, the action “Email\_sending” uses “CUSTOMER\_NAME” and “PRODUCT\_ID” as arguments for campaign contents, and defines the target data of campaign analysis: “READ\_TIME” (when the customer reads the Email), “READ\_FREQUENCY” (how often the customer reads the Email), and “IS\_REJECT” (whether the customer rejects the Email or not). Throughout the paper, actions mean campaign actions.

Now, with the above two definitions, we define the campaign rules.

**Definition 3 (Campaign rules)** A campaign rule is defined as **IF** \( \{\text{term} \mid \text{value}\} \{\text{or} \text{term} \mid \text{value}\} \) **THEN** \( \{\text{campaign action}\} \) where \( \text{op} \) is the relational operator such as ‘\( = \)’, ‘\( \neq \)’, ‘\( \geq \)’, ‘\( \leq \)’ and ‘\( \neq \)’.

As seen in the definition, the **IF-clause** is represented as a conjunctive normal form (CNF) or disjunctive normal form (DNF) to determine a group of target customers.

**Example 3** Suppose that a marketer attempts to promote the sales of a new product to customers who are in the twenties and whose purchase capacity for the last month exceeded $100 by Email. The campaign can be defined as the following rule.

\[
\text{IF} \{\text{PURCHASE\_CAPACITY\_OF\_LAST\_MONTH} \geq 100 \text{ AND} \text{AGE\_GROUP}=20 \} \text{ THEN} \{\text{"Email\_sending"}, \{\text{"CUSTOMER\_NAME"}, \{\text{"READ\_TIME"}, \text{"READ\_FREQUENCY"}, \text{"IS\_REJECT"\}}\}\}
\]

where the meaning of terms in the measures part is the same as that of terms used in Example 1.

Now, we consider the necessary condition for composing valid campaign rules.

**Condition 1** A term \( v \) in **IF-clause** of a given campaign rule should satisfy one of the following conditions.

1. The subject of \( v \) is the primary key (denoted as \( \text{customer\_id} \)) of customer table.
2. Among terms \( v_1, \ldots, v_n \) which are connected as **AND** operator with \( v \), there is a list of terms \( \{v_i, v_j, \ldots, v_k\} (1 \leq i, j, k \leq n) \) such that \( v\_i\_\text{subject}=v\_i\_\text{link}, v\_i\_\text{subject}=v\_i\_\text{link}, \ldots, v\_j\_\text{subject}=v\_j\_\text{link}, \) and \( v\_k\_\text{subject}=\text{customer\_id} \).
In order for a given campaign rule to be valid, some terms within the IF-clause of the rule should address attributes directly associated with customers while others should constrain these attributes. In other words, for a given term, its subject should be a primary key of customer table, or it should be connected as AND operator with a list of terms where the link of one term is equal to the subject of the next connected term.

4 Building rule-aware data warehouse configuration

4.1 Schema generation

In order to generate a data warehouse schema corresponding to a given campaign rule, we use the subject/link information of terms within the rule and the measures information within the THEN-clause. The overall procedure is as follows.

1. Determine the target of analysis that is stored in fact tables from measures part of a given campaign action (see Section 4.1.2).
2. Generate a schema of dimension tables using subject/link information of terms within a given rule (see Section 4.1.1).
3. Make up composite keys (a set of foreign keys referencing primary keys of dimension tables) of fact tables (see Section 4.1.2).

The key point to the proposed algorithm is to generate dimension tables. The schema of dimension tables can be generated as follows.

4.1.1 Generation of dimension tables

1. Define a set of terms ANALYSIS_TERMS that can analyze the information of measures part of the campaign action from operational data sources. Such terms are determined with attributes existing in the table in which the primary key equals the subject of a term in the measures part.
2. Compute a set of terms DIM_TERMS that contains terms occurring in the IF-clause of a given campaign rule and combine it with ANALYSIS_TERMS obtained in step 1.
   \[ \text{DIM_TERMS} = \text{ANALYSIS_TERMS} \cup \{ v | v \text{ is a term occurring in IF-clause} \} \]
3. Generate partition \( V_i \subset \text{DIM_TERMS} \) by subject of terms belonging to the set \( \text{DIM_TERMS} \).
   \[
   V_i = \{ v | v \in \text{DIM_TERMS} \text{ and the subject of } v \text{ is } i \ (1 \leq i \leq |\text{subject}|, \text{where } |\text{subject}| \text{ is the number of distinct subjects of terms in } \text{DIM_TERMS} \}\]
4. Generate partition \( d_{ij} \subset \text{DIM_TERMS} \) by link information of each term belonging to \( V_i \).
   \[
   d_{ij} = \{ d | d \in V_i \text{ and the link of } d \text{ is } j, \text{where } 1 \leq j \leq |\text{link}| \} \}
   \[
   |\text{link}| \text{ is the number of distinct links in terms belonging to } V_i \}
5. For each term \( v \in d_{ij} \), check the value of link, and generate \( d_{ij}^{\text{null}} \), which is a set of terms whose value of link is null, and \( d_{ij}^{\text{null}} \) which is a set of terms whose value of link is not null.
6. For each \( d_{ij}^{\text{null}} \), generate a dimension table consisting of a primary key being the subject \( i \) and fields corresponding to all terms belonging to \( d_{ij}^{\text{null}} \).
7. For each term \( v \) belonging to \( d_{ij}^{\text{null}} \), generate a dimension table \( T_v \) as follows. \( T_v \) consists of the foreign keys corresponding to \( v \)'s subject and link and the field whose name is the term \( v \) itself.
8. Create (or update) a ‘time’ dimension table consistent with a generated fact table.

Note that even though certain fields do not appear in a given rule, they can be used as dimensions for analyzing the measures information (see step 1). All of the possible fields that can belong to the dimension tables are extracted (see step 2), and they are then decomposed into several partitions by subjects of terms (see step 3). Each partition of the fields is decomposed into more refined partitions by using link information of terms (see step 4). Then, by checking the value of link, each partition is again divided into two groups: \( d_{ij}^{\text{null}} \) and \( d_{ij}^{\text{null}} \) (see step 5). The terms belonging to \( d_{ij}^{\text{null}} \) make up a dimension table with each subject \( i \) since \( d_{ij}^{\text{null}} \) of a certain subject \( i \) includes terms which are assumed to have the identical viewpoint of analysis (see step 6). In contrast, each of the terms belonging to \( d_{ij}^{\text{null}} \) composes a separate dimension table so that multivalued dependencies can hold on each of generated tables (see step 7). If all terms relevant to \( d_{ij}^{\text{null}} \) build up a single table as in \( d_{ij}^{\text{null}} \), we actually have less information since we obtain not only the records we had originally, but also several additional records.

Since a ‘time’ dimension table, which contains information of time, is mandatory in designating a data warehouse, we need to prepare time relevant fields such as year, month, day, and holiday. This dimension table is manually constructed so that it is consistent with the generated fact tables in terms of their granularity. Thus, step 8 should be executed after generating (or updating) fact tables.

4.1.2 Generation of fact tables

Basically, for a given campaign rule, only customer ID and MEASURES (that includes terms to be target of analysis) are enough for its corresponding fact table. This is because terms whose subject is not customer ID can be joined with customer ID by Condition 1. Moreover, only if fact tables has customer ID as a foreign key, they can be analyzed by fields belonging to ANALYSIS_TERMS. However, such a fact table with only the least information yields complex analysis queries using foreign key constraints. Consequently, the queries yields excessive execution time
since they require many join operations between related tables. Hence, we allow the schema of fact table to include ANALYSIS_TERMS that includes terms not appearing in a given rule even though the schema has redundant information. Consequently, the fact table FACT has the following schema.

$$\text{FACT} = \text{MEASURES} \cup \{\text{CUSTOMER_ID}\} \cup \{FK\} \quad \text{FK is an attribute that is the subject of each of rule terms belonging to ANALYSIS_TERMS}$$

where FK is a foreign key referencing the primary key in its corresponding dimension table.

4.2 Indexing constraint set-up and analysis query generation

In the proposed method, the fields related to subject (or link) information in dimension tables are ones that can be joined with fact tables or other dimension tables. Thus, we can create star-join indexes by correlating the value of such fields with foreign keys within fact tables beforehand. In addition, for fields other than subject and link, B-tree indexes are generated in advance.

As for analysis queries, since the proposed method generates currently needed data schema based on given campaign rules, all possible analysis queries over the generated dimensions and facts are generated. Typically, analysis queries are described in SQL as follows.

```
SELECT grouping fields, aggregation functions (parameters)
FROM fact tables, dimension tables
WHERE join constraints
GROUP BY grouping fields
ORDER BY grouping fields
```

In SELECT-clause, the aggregation function is applied to all fields related to the measures part of action. The grouping fields belong to all fields within the dimension tables. In WHERE-clause, join constraints represent a set of join operations using foreign key constraints between fact tables and dimension tables. In addition, by browsing the dimension tables, users may add some application constraints that serves to restrict a certain dimension to only interested records on top of the generated queries.

Since these analysis queries can be generated beforehand, we can obtain their optimized query plans before executing the queries, as if canned queries can be compiled ahead of execution time. Even though the proposed method generates too many analysis queries, users can easily choose the best queries that they want only if a user-friendly user interface is provided.

5 Examples

Suppose that marketers attempts to promote new products for two types of customers by performing direct Email services. Then, they define their campaign strategy as campaign rules of Figure 3. Moreover, after executing the Email marketing, they want to analyze the response results of the customers.

We assume that the operational data sources are defined as the following relational schema. Each of the attributes indicated by underline serves as a primary key.

- Customers (customer_id, name, sex, zip_code, address, job)
- Region (zip_code, city, street)
- Response (response_id, mail_id, customer_id, is_read, read_begin_time, read_end_time, sales_id)
- Product (product_id, brand, category, weight, color)
- Sales (sales_id, customer_id, product_id, time, sales_amount)
- Mailer (mail_id, title, contents, send_time, num_of_customers)
- Purchase (purchase_id, customer_id, product_id, date)
- Email (email_id, mail_id)
- Region (zip_code, city, street)
- Measurement (product_id, category, value)
- Age (age)
- City (city)
- Product (product_id, brand, category, weight, color)
- Sales (sales_id, customer_id, product_id, time, sales_amount)
- Mailer (mail_id, title, contents, send_time, num_of_customers)
- Purchase (purchase_id, customer_id, product_id, date)
- Email (email_id, mail_id)
- Region (zip_code, city, street)
- Measurement (product_id, category, value)
- Age (age)
- City (city)

Figure 3: Examples of campaign rules

- Rule1

<table>
<thead>
<tr>
<th>IF</th>
<th>THEN</th>
</tr>
</thead>
</table>
| IF AGE_GROUP = 20 AND city='Seoul' Purchase_capacity > $500 AND purchase_item = 'Computer' AND category = 'Server' AND quantity_sold > 100,000 | THEN ("Email_sending", 
\{"customer_name", 
"recommended_products"}, 
\{"is_read", "read_duration", "Purchase_capacity"\}) |

- Rule2

<table>
<thead>
<tr>
<th>IF</th>
<th>THEN</th>
</tr>
</thead>
</table>
| IF sex = 'female' AND mileage > 1000 AND Purchase_capacity < $500 AND purchase_items_of_last_month = 'Computer' AND product_date > '1/1/2000' | THEN ("Email_sending", 
\{"customer_name", 
"recommended_products"}, 
\{"is_read", "read_duration", "Purchase_capacity"\}) |

5.1 Definition of rules terms

Rule terms used in campaign rules of Figure 3 are defined as follows.

- **PURCHASE_ITEMS_OF_LAST_MONTH**
  
  $$\text{SELECT c.customer_id, s.product_id FROM Customer c, Sales s WHERE c.customer_id = s.customer_id AND s.purchase_year = \text{EXTRACT}(\text{Year from Current_date()}) AND s.purchase_month = \text{EXTRACT}(\text{Month from Current_date()})-1 \text{GROUP BY c.customer_id"}$$

- **PURCHASE_CAPACITY**
  
  $$\text{SELECT c.customer_id, sum(s.amount) FROM Customer c, Sales s WHERE c.customer_id = s.customer_id GROUP BY c.customer_id"}$$

- **AGE_GROUP**
  
  $$\text{SELECT c.customer_id, Extract(\text{Year from (birthdate-Current_date())}/10+10 FROM Customer")}$$

- **CITY**
  
  $$\text{SELECT c.customer_id, r.zip_code, c.region FROM Customer c, Region r WHERE c.zip_code = r.zip_code"}$$
5.2 Generation of star schema

Given the campaign rules in Figure 3, the proposed method generates a data warehouse schema shown in Figures 4 & 5.

For the two campaign rules Rule1 and Rule2, the variables used in the schema generation algorithm of Section 4.1.1 have the following results.

Rule1
- \( \text{ANALYSIS_TERMS} = \{ \text{name, sex, address, job} \} \)
- \( \text{DIM_TERMS} = \{ \text{name, sex, address, job} \} \cup \{ \text{age_group, city, purchase_capacity, purchase_item, category, quantity_sold} \} \)
- \( V_{\text{customer}_{id}} = \{ \text{name, sex, address, job, age_group, city, purchase_capacity, purchase_item} \} \)
- \( V_{\text{product}_{id}} = \{ \text{category, quantity_sold} \} \)
- \( d_{(\text{customer}_{id}, \text{product}_{id})} = \{ \text{purchase_item} \} \)
- \( d_{(\text{customer}_{id}, \text{NULL})} = \{ \text{age_group, city, name, sex, address, job, purchase_capacity} \} \)
- \( d_{(\text{product}_{id}, \text{NULL})} = \{ \text{category, quantity_sold} \} \)

Rule2
- \( \text{ANALYSIS_TERMS} = \{ \text{name, sex, address, job, brand, category, weight, color} \} \)
- \( \text{DIM_TERMS} = \{ \text{name, sex, address, job, brand, category, weight, color} \} \cup \{ \text{sale, mileage, purchase_capacity, purchase_item, \text{of last month}, product_date} \} \)
- \( V_{\text{customer}_{id}} = \{ \text{name, sex, address, job, brand, category, weight, color} \} \)
- \( V_{\text{product}_{id}} = \{ \text{category, quantity_sold} \} \)
- \( d_{(\text{customer}_{id}, \text{product}_{id})} = \{ \text{purchase_item} \} \)
- \( d_{(\text{customer}_{id}, \text{NULL})} = \{ \text{brand, category, weight, color, \text{of last month}, product_date} \} \)
- \( d_{(\text{product}_{id}, \text{NULL})} = \{ \text{brand, category, weight, color, \text{of last month}, product_date} \} \)

For given rules Rule1 (or Rule2), \( \text{RULE1\_FACT} \) (or \( \text{RULE2\_FACT} \)) fact table is generated from measures information of the rule, and four dimension tables such as CUSTOMER, PRODUCT, PURCHASE_CAPACITY and \( \text{PRODUCT\_ITEM} \) (or \( \text{PRODUCT\_ITEM\_OF\_LAST\_MONTH} \)) are generated by using \textit{subject/link} information. The CUSTOMER and PRODUCT dimension tables are generated from \( d_{(\text{customer}_{id}, \text{product}_{id})} \) and \( d_{(\text{customer}_{id}, \text{NULL})} \), respectively. The \( \text{PRODUCT\_CAPACITY} \) and \( \text{PURCHASE\_ITEM} \) (or \( \text{PRODUCT\_ITEM\_OF\_LAST\_MONTH} \)) dimension tables are generated from \( d_{(\text{product}_{id}, \text{NULL})} \) and \( d_{(\text{customer}_{id}, \text{product}_{id})} \), respectively. The two CUSTOMER (or PRODUCT) dimension tables generated from Rule1 and Rule2 can be combined into a single table if one table subsumes the other table. To do this, the subsumption needs to be determined by checking selection conditions in their related SQL statements.

By doing so, the schema automatically generated from rule information evolves into a galaxy-style
schema that has several fact tables sharing dimension tables while processing different campaigns.

6 Related work

Research work on automatically creating data warehouse configuration from external user requirements was carried out only recently in (Lee et al. 2000). In that work, a user can graphically browse the data by the result of representing his or her thoughts, and by using catalog information, he or she can generate database schema for OLAP. The related work deals with designing materialized views by utilizing frequently asked questions.

In terms of designing optimal materialized views, (Baralis et al. 1997) proposed an algorithm that reduces search space for searching for materialized views by using heuristics information such as functional dependency, hierarchical structure, and the size of materialized views. In (Theodoratos et al. 1997), when a submitted query is processed only with materialized views, a method of minimizing the maintenance cost is proposed. (Shukla et al. 1998) improved the algorithm for selecting materialized view of a special cell with limited size in the literature of data cube. Recently, (Agrawal et al. 2001) proposed a method that designs materialized views and indexing constraint with Microsoft SQL Server. This enables to provide a materialized view by pruning the search space of materialized view and merging two materialized views. However, such materialized views require joining the same relations. In (Chirkova et al. 2001), according to completeness of cost model view, complexity of such a selection problem was theoretically analyzed.

7 Conclusions

We have presented a novel method of automatically developing data warehouse configuration in rule-based CRM systems. The proposed method can significantly reduce cost incurred in building and maintaining data warehouse. Since the proposed method enables the system to store only necessary information in data warehouse, the data warehouse can be highly utilized. Also, the method can effectively maintain data warehouse configuration while reflecting new marketing campaigns over time. We believe that our present approach significantly will help to build analytical CRM systems with data mining applications in that the problem of building and maintaining useful data warehouses remains to be one of the greatest obstacles to successful data mining. As data warehousing is a greatly risky task that requires a large amount of money and time, many data warehouse experts have suggested a motto, “Think big but start small”, to data warehouse developers. And, the proposed method fits such a motto.

We are still the process of investigating techniques for normalizing automatically generated schema in order to avoid data redundancy. The basic idea is that two dimension tables such that their subjects are the same and their links are null can be combined into a single table through SQL union operation.

References


