A DESIGN OF INTELLIGENT PRE-FETCHING MATERIALIZED VIEWS MECHANISM FOR ENHANCING SUMMARY QUERIES ON DATA WAREHOUSES

Chin-Feng Lee and Main-Che Tsai
Department of Information Management, Chaoyang University of Technology
No. 168, Gifeng E.Rd., Wufeng, Taichung County, Taiwan 413
EMAIL{lcf, s8854606}@mail.cyut.edu.tw

ABSTRACT

To build up a materialized view that perfectly satisfies the need of the specific enterprise it serves is now the biggest challenge especially when it comes to larger and larger scale enterprises as well as more and more complicated and yet necessary socio-economical information. In this paper, we shall develop an Intelligent Materialized VIews Pre-fetching mechanism, also known as an IMVIP, from the characteristics of affinity grouping so as to enhance the efficiency of summary data warehouse querying.

The IMVIP mechanism consists of the following two methods: the Apriori-Model association method and the Linear Structure Relation. The Apriori-Model association method explores and deduces the combination of the relations among individual user session. It is especially suitable for applications where the combinations of the relations are to be explored among multi-objective queries made by more than one decision maker. On the other hand, the Linear Structure Relation Model develops a set of principles as to the explorations into the deduced relation combination above with an aim to constructing a series of causal-effect association rules. Thus, we can not only pre-fetch and materialize views that really satisfy the needs of the decision makers so as to enhance the efficiency of summary data warehouse queries but also build up intelligent query paths according to the cause-and-effect association rules in order to attain the goal of providing helpful suggestions for decision-making.

Keywords: data warehouse, materialized view, data mining, association rule, linear structure relation model.

1. INTRODUCTION

As the business management environment becomes more and more complicated, bunches and bunches of different data sources are to be consulted during the decision-making process in order to make the decision more deliberately considered. However, one of the most important problems we must face up to is that the places where the data are stored are most probably widely distributed, divergent, and heterogeneous. To solve this problem, such a design as the summary table manageable mechanism, or namely view materialization, has been implemented in decision supporting systems or data warehouses [9][10][12][13]. This way, the information needed for decision-making can be filtered and combined beforehand at the data sources, and the summarized data can be offered by functions such as total (sum), average, or count.

At the first glance, it looks as if the interconnections between materialized views and data sources could solve the above problem; however, in reality, it is whether the materialized view constructed can meet the requirement of the query for decision-making or not that is the key point that determines whether or not the problem above can be successfully solved. Therefore, how to build up materialized views and make sure that the contents of the materialized views constructed can really satisfy the needs is the biggest challenge for all enterprises at the present time. In addition, in the multi-user environment, a decision maker would often have to make a series of queries into the data warehouses for one single decision, where the collection of the series of queries is called a "user session". Queries for decision-making are races against time, and therefore, to profit more in the tough competitions nowadays, shortening the time consumed to answer queries and speeding up the acquisition of summary information are tasks that cannot wait. In this paper, making use of the characteristics of affinity grouping among strings of summary queries, we shall develop an intelligent materialized views pre-fetching mechanism, also known as an IMVIP, to enhance the efficiency of summary data warehouse querying.

The mechanism that we plan to develop here is based on the very feature of views that they are purely customized. Finding out the angles from which users dig into the data, keeping record of the data utilization, we shall propose the Apriori-Model, which is a revised association method of Aprioriall [1] of higher efficiency, to dig into the association combinations between the individual user sessions and the categories that the data referenced fall into. With the help of the pre-fetching mechanism, we can effectively cut down the occurrence of situations where the target information falls off the data warehouses and has to be found in the jungle outside. On the other hand, if we can take a step further and establish the causal relations among the association combinations, then we can succeed in providing data warehouse users with an intelligent information offering environment. Therefore, in this paper, we shall adopt the linear structure relation (LISREL)[7] to develop a set of principles that govern the digging into the association combinations by means of building up the cause-and-effect association rules. After the two stages of data mining above, we can not only pre-fetch as well as materialize the views the decision makers are really in need of, but also establish intelligent query paths according to the behavioral patterns of the decision makers and the causal relations and hierarchical ranking of the data referenced.

The rest of this paper is organized as follows. In Section 2, we shall explore into the literature concerned to have a clear picture of what steps researchers have made in the history of this field of study. Section 3 will portray the process in which we translate the records of data utilization into the data structure we need. Then, in Section 4, we shall discuss the methodology and steps in our procedure. Here, we shall give a detailed description as to how we can come up to the association combinations by means of the Apriori-

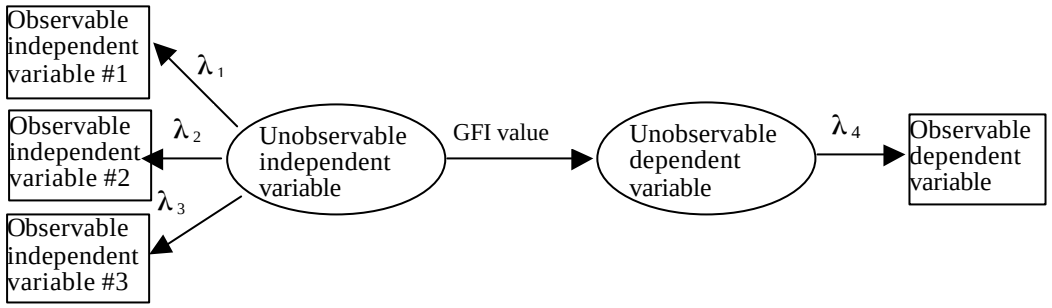


Figure 1. Part of the causal relationships in our research

Model to avoid the weakness of AprioriAll in repeatedly scanning the data warehouses. Besides that, we shall also describe how we derive the causality of the association combinations from the digging principles we develop out of the linear structure relation model. Then, in Section 5, we shall discuss the practicability of the association combinations with causality we come up with in real applications. Finally, the conclusions will be in Section 6.

2. LITERATURE REVIEWS

The metadata is an important element of the data warehouses environment. The term "metadata" refers to the descriptive information of the data in the data warehouses. It has multiple functions, among which is the establishment of the rules for data association and access [3]. Therefore, the rules for causality-attached relations that we derive in this paper can be stored in metadata as criteria for the establishment of materialized views. However, little has been discussed as to how to evaluate whether the contents of materialized views can successfully cover the needs of decision makers as well as how to build up a pre-fetching mechanism to speed up the decision-making process. These two untouched problems are right the points of this paper.

Deducing association rules for the decisions to be made in different areas and different data structures can be very helpful for sequence patterns [1], for multiple-level data [5], and for weighted items [4]. In addition, the establishment of the web searching engine with text mining method [8], and the restoration of mining values via association rules [11] are all effective ways to help settling on business strategies and action plans. To join the club, in this paper, we shall propose a revised version of the Aprioriall association method [1], named the Apriori-Model. Based on the strong association among individual sessions under the same decision to be made, the Apriori-Model is indeed more effective in deriving association rules.

To measure the interrelationships among the factors affecting decision-making, statistics proves to be a good choice. The linear structure relation (LISREL) is a good statistical method to turn causal relations among variables into numbers [2][7]. In this paper, we shall make use of LISREL to derive the causality of the association combinations. As we mentioned earlier, LISREL is basically an analytical method to determine if a linear relation exists between two factors that are not directly observable. Such a model can help determine if there exists a causal relationship between the unobservable dependent variable (derived from the observable dependent variables) and the unobservable independent variable (derived from the observable independent variables). As Figure 1 shows, the goodness of fit indicator (GFI) is used to measure the fitness of this causality model. The GFI value can be anywhere between 0 and 1, and a greater GFI value means a higher degree of fitness. The GFI threshold is determined according to professional need for different subjects. The symbol ë stands for every unobservable independent (or dependent) variable to observable independent (or dependent) variable regression coefficient (magnitude of influence). In this paper, there is only one observable dependent variable; in other words, $ë_4$ is equal to one. On the other hand, there is more than one observable independent variable in our research. Nevertheless, all the observable independent variables cannot be included in the relation model. Only when the evaluation result of a "cause" reveals that the p-value is less than 0.05 does it mean that this cause has a significant impact on the effect and that this cause is to be taken as an element of the association rule.

3. DEFINITION OF DATA STRUCTURE

To make the materialized views of data warehouses meet the needs of decision makers and thus to enhance the efficiency of data access as well as to reduce the load of the network and the frequency of data access in heterogeneous, distributed data warehouses, in this paper, we shall build up association rules of causal relationships for the data queried by decision makers. Based on the Apriori-Model and the linear structure relation, the association rules are derived from the interrelations among user sessions. That is to say, according to the behavioral patterns of the decision makers revealed by the queries, we can deduce rules that can not only serve as guides of data withdrawal for every decision maker but also enhance the effective use of data resources if the association rules derived are incorporated in the data warehouses.

Back in the first section, we have defined a user session as the collection of a series of queries made by one decision maker for one decision to be made in some multiple-user environment. Here in this place, we shall take one step further. The queries made for one decision are defined as the string of queries a decision maker makes from the moment she/he enters the data warehouse (or is led by the warehouse to some other databases if the data warehouse lacks the data wanted) until when she/he leaves. Each and every query can be recorded in the data structure shown in the first part of Figure 2. Then, the series of queries made for one decision (namely one user session) can be organized as the second part of Figure 2 shows, where the session code is the sequential number of the user session and the code for data queried is composed of the data codes

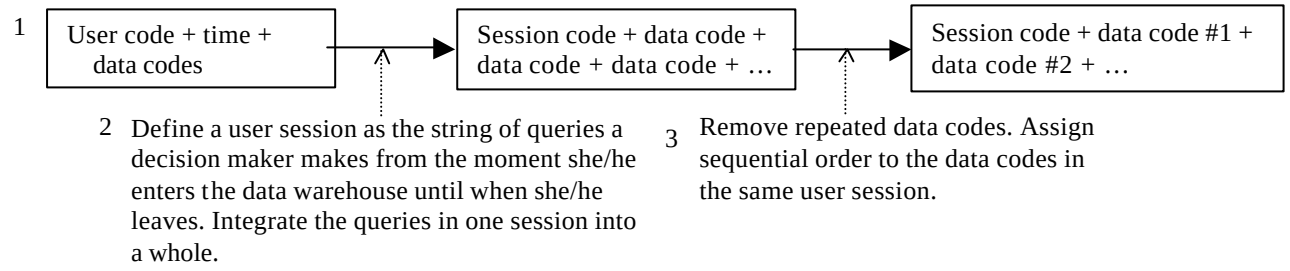


Figure 2. Procedure of integrating user sessions into data structure

of a series of queries.

In addition, there are still two features to the data that we keep in store. Number one, a decision maker may probably ask for data of an especially wide variety in her/his one user session. For example, in the series of queries (sql 1, sql 2, sql 3, sql 4), suppose query sql 1 is made for "population density," query sql 2 for "crime rate," query sql 3 for "income," and query sql 4 is made for "salary." In this place, the sequential order of the queries made is decided at random; in other words, for the decision maker, there is no sequential order to the queries. Although sometimes the sequential order might actually mean something to the decision maker, we decide to ignore it here. To give an example, suppose a user wants to analysis the "crime rate." She/he makes queries in the arbitrarily decided order of "population density," "income," and then "salary;" namely, the queried items are only weighted according to their expected magnitude of influence on the "crime rate" in such order subjectively by the user. However, in the rest of paper for simplify the process, we can give sequential order to the data codes belonging to the same session code (see the third part of Figure 2). Second, there can be no repetition in one user session. In other words, the "population density," for example, is unlikely to be queried twice in one user session because there is simply no meaning to it (also see the third part of Figure 2).

4. THE METHODOLOGY AND PROCEDURE

To gather the information needed to make one decision, a decision maker must make a string of queries in a user session. For a same decision to be made, the user sessions of different decision makers must bear certain relativity and similarity with each other. The major objective of this paper is to derive the behavioral patterns of query makers from the data access records and thus to design an intelligent materialized views pre-fetching mechanism. In practice, such a mechanism can materialize views for data warehouse users beforehand, or the materialized views can be pre-fetched and stored in memory or buffer memory so as to save the time wasted on gathering widely distributed data and to relieve the load of the networks. Thus, the efficiency of data querying can be enhanced. Besides, up to now, data mining techniques have been widely used. Under such circumstances, if we can analyze the behavioral patterns of the users and explore the goals of the data queries they make, establishing the causal relations among the elements in the association combinations via the linear structure relation, then we can weight the causes and effects accordingly, giving them their query order, and generate an intelligent instructive structure to guide data warehouse users through the querying process. Thus, the efficiency of data warehouse operation can be dramatically raised.

Take Figure 3 for example. According to the data structure derived in Section 3, we can put the meaning into the codes in Figure 3. Thus, the series of queries (sql 1, sql 3, sql 5, sql 8) can be considered the SQL queries some certain user makes for "population" (sql 1), "average income per capita" (sql 3), "land area" (sql 5), and "average tax paid per capita" (sql 8). By means of association mining, we can build up the association combination among the three {the query for "population" (sql 1), the query for "average income per capita" (sql 3), & the query for "land area" (sql 5)} (see the second part of Figure 3). Then, with the help of the linear structure relation, we come to the conclusion that there exists a causal relationship in the string of the query for "population" (sql 1), the query for "land area" (sql 5), and the query for "average income per capita" (sql 3) (see the third part of Figure 3). Finally, we can figure out that the query for "land area" (sql 5) is weighted heavier than the query for "population" (sql 1). Therefore, the final causal relationship goes that the query for "land area" (sql 5), the query for "population" (sql 1) -> the query for "average tax paid per capita" (sql 3) (see the fourth part of Figure 3). Such a rule can be stored in the metadata. This way, we can put together a rule-based engine organized by causality rules. Users can then profit from the materialized views pre-fetching rules and the intelligent instructive mechanism for querying.

To establish association rules of causal relations that the materialized views can base themselves upon in order to

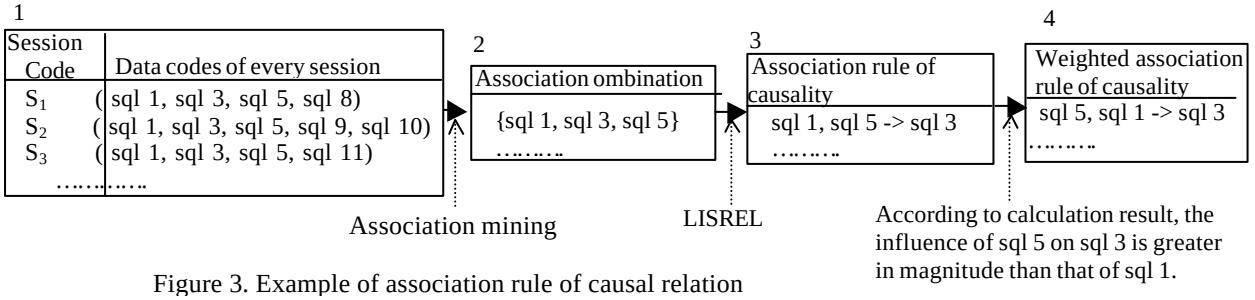


Figure 3. Example of association rule of causal relation

provide decision makers with intelligent guides, we take the following five steps:

Step 1. Establishing Decision Models

When the queries are asked for the same objective by a number of decision makers different from each other, the data accessed must have considerable similarity. User sessions that are similar to each other can then be organized into a decision model. This way, we can roughly classify sessions whose objectives cannot be told at the first glance. Such association rule establishment upon similar sessions is obviously more meaningful than that built upon all sessions with mixed objectives. For example, to explore the increase/decrease rate of population in a certain district, decision maker A might make queries for "marriage rate" (sql_i), "migration rate" (sql_j), as well as "birth rate" (sql_r). That is to say, the user session for decision maker A is S_A = (sql_i, sql_j, sql_r). For another decision maker B, the three items above might not be enough. Another query might be made for "unemployment rate" (sql_t). Namely, the user session for decision maker B is $S_B = S_A \cup \{(sql_t)\}$. These two very much similar sessions can be organized into a decision model $S_A \cup S_B$ = {(sql_i, sql_j, sql_r), (sql_i, sql_j, sql_r, sql_t)}.

Here, we employ a formula to figure out the probability of overlapping for two events that goes $P(A \cap B) = n(A \cap B) / n(A \cup B)$ [6] to help decide if two sessions are similar to each other. In the formula, the item $n(A \cap B)$ stands for the number of elements occurring in both events A and B. On the other hand, the item $n(A \cup B)$ represents the total number of the elements included in the two events, where an element occurring in both events can only be counted once; in other words, $n(A \cup B) = n(A) + n(B) - n(A \cap B)$. In this paper, we define the term "model affinity" as the degree of resemblance between the two events A and B. When testing the model affinity, the decision maker has to offer a threshold to filter the events and decide whether they should be put into the decision model. Such a threshold is given the name of "minimum model affinity" in this paper.

In the process of decision model construction, one session S_i is picked out to be the primary session at first, and the other sessions, named compared sessions, are extracted one by one to compare with the primary session and decide the model affinity, which is represented by $ma(S_i, S_j)$. Here, the same sqls between S_i and S_j can be collected together and represented by $same(S_i, S_j)$. When the value of $ma(S_i, S_j)$ turns out to be higher than the minimum model affinity ($\hat{\imath}$) preset by the decision maker, then, by definition, S_i and S_j are declared to be similar to each other, and S_j is put into the decision model $DM(S_i)|_{\hat{\imath}}$ where S_i is the primary session. For example, the total number of sessions in Figure 4 is N = 5. The following are the steps of decision model construction.

Session code	Query codes for every session
S_1	(sql 1, sql 2, sql 3, sql 4)
S_2	(sql 1, sql 3, sql 4, sql 5)
S_3	(sql 1, sql 2, sql 3, sql 6, sql 7)
S_4	(sql 1, sql3, sql 5)
S_5	(sql 4, sql 5)

Fig. 4 User sessions

Step 1-1. Take S_1 for the primary session (suppose $\hat{\imath}$ = 50%).

(1) Take S_2 for the compared session.
Since session S_1 and session S_2 have three queries in common, namely $same(S_1, S_2)$ = {sql 1, sql 3, sql 4}, the model affinity between S_1 and S_2, namely $ma(S_1, S_2)$, is: number of queries S_1 and S_2 have in common / total number of queries in sessions S_1 and S_2 (repetitions eliminated) = 3 / 5 = 60%.

(2) Take S_3 for the compared session and obtain $ma(S_1, S_3)$ = 3 / 6 = 50%.

(3) Take S_4 for the compared session and obtain $ma(S_1, S_4)$ = 2 / 5 = 40%.

(4) Take S_5 for the compared session and obtain $ma(S_1, S_5)$ = 1 / 5 = 20%.

Learning from (1) through (4) of Step 1-1, when S_1 acts as the primary session, we take S_2, S_3, S_4, S_5, as well as S_1 out one by one to compare them with the primary session. Among them, S_2 and S_3 are more similar to S_1 because both of their model affinity values are larger than $\hat{\imath}$ (in other words, $ma(S_1, S_2) > 50\%$ and $ma(S_1, S_3) > 50\%$). Therefore, we can establish the decision model whose primary session is S_1 that goes $DM(S_1)|_{\hat{\imath}=50\%}$ = {S_1, S_2, S_3}.

Step 1-2. By the same token, we can establish a decision model whose primary session is S_2 as $DM(S_2)|_{\hat{\imath}=50\%}$ = {S_1, S_2, S_4, S_5}. Then, based on another primary session S_3, we can establish the decision model $DM(S_3)|_{\hat{\imath}=50\%}$ = {S_1, S_3}; for primary session S_4, the decision model is $DM(S_4)|_{\hat{\imath}=50\%}$ = {S_2, S_4}. Finally, for primary session S_5, the decision model is $DM(S_5)|_{\hat{\imath}=50\%}$ = {S_2, S_5}.

Step 2. Pruning Subset Decision Models

Some of the decision models we come up with might turn out to be the subsets of other decision models. For two decision models where one is the subset of the other, they can be considered a long sequence queries for on purpose and treated as a whole decision model. Cutting off the redundancy, we remove the smaller decision model (also called the subset decision model). For example, $DM(S_3)$ is a subset of $DM(S_1)$ and should thus be eliminated; at the same time, $DM(S_4)$ and $DM(S_5)$ are both subsets of $DM(S_2)$, and thus they should both be removed. For now, only $DM(S_1)$ and $DM(S_2)$ survive.

Step 3. Filtering out Non-Significant Decision Models

When a decision model has been established, we should then consider its popularity among all the sessions. In other words, if a decision model seldom appears, then it is non-significant to the needs of decision makers. Therefore, such a decision model of low popularity is considered not worthy of exploring and is thus eliminated. In this paper, we define the session support of S_i as $ss(S_i) = |DM(S_i)| / N$, where $|DM(S_i)|$ stands for the number of sessions whose decision models are based on the primary session S_i, and N stands for the total number of the sessions. When the value of session support is higher than the minimum session support (β), it means the decision model is significant. In our example, the session support value of $DM(S_1)$ is $ss(S_1)$ = 3/5 = 60%, and the session support of $DM(S_2)$ is $ss(S_2)$ = 4/5 = 80%. Suppose the preset minimum session support is 70%, then $DM(S_1)$ should be eliminated. Therefore, for now, only $DM(S_2)$ survives.

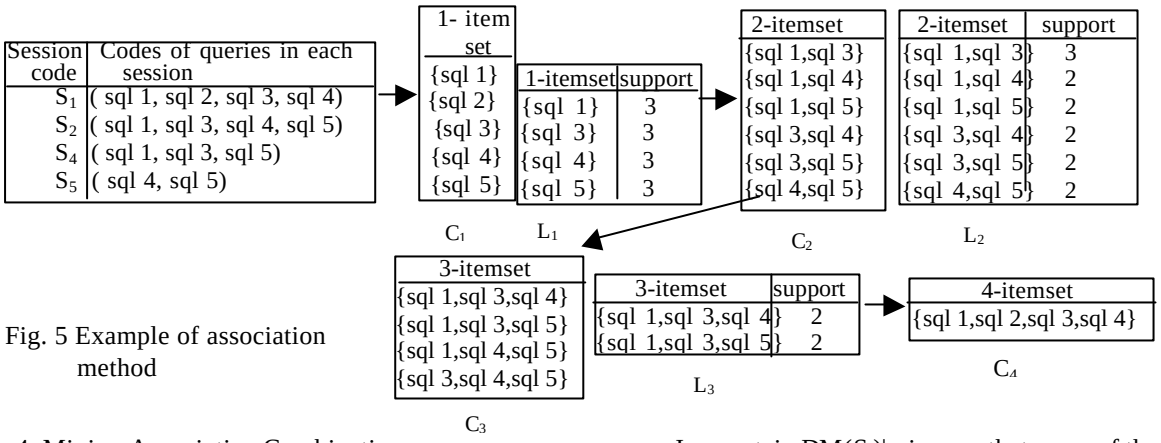


Fig. 5 Example of association method

Step 4. Mining Association Combinations

Now, we can start to build up association combinations among the sessions in every decision model $DM(S_i)$ by association method. Here, we shall first describe how to construct association combinations by the Aprioriall association method [1]. Then, to enhance the efficiency, we shall propose the Apriori-Model to get rid of Aprioriall's weakness of repeatedly scanning the databases. In this paper, we define "model support" as the probability of occurrence of the query for a certain data item in different sessions of the same decision model. To filter data items and decide if they can be admitted in the association combination, a minimum model support value (represented by the symbol á) is set for comparison.

Take Figure 5 for example (where the decision model is $DM(S_2)$). Following the Aprioriall association method, we come from C_i through L_1 and C_2 to L_2, and so on and so forth. Here, C_i is the candidate set of i-itemset's, and L_i is the set composed of i-itemset's that have the property of Large, meaning that the model support values of these i-itemset's are greater than the minimum model support. In this example, we set the minimum model support to be 50%. Besides, to make the association combinations in Li (i = 1, 2, 3 and 4) acceptable, they must be maximal. Here, an association combination that is maximal cannot be a subset of another association combination.

When deriving decision models $DM(S_i)$ empirically, what we need are decision models that bear high resemblance among them so that there can be high similarities among sessions, which in turn means a convergent objective. Decision models sorted out by a higher minimum model affinity value have sessions of higher resemblance. Therefore, association combinations can be more easily derived from the common queries among sessions, namely $same(S_i, S_j)$ instead of following the Aprioriall association method through the long way from 1-itemset, 2-itemset, …. to n-itemset.According to the discussion above, we propose the Apriori-Model association method here to derive association combinations with the property of "Large" from $same(S_i, S_j)$. The procedure is described as follows:

[P1] Sort out sessions with the property of "Large" from $DM(S_i)|_{\hat{i}}$

When the minimum model support is set to be d%, we can use the equation N × d% = f to turn the threshold value from the form of a percentage (d%) into a number (f). This number of appearances f is the lowest point that the itemset must reach to demonstrate it is "large" enough. This lowest f is entitled the minimum model support count.

In a certain $DM(S_i)|_{\hat{i}}$, in case that some of the sessions have identical contents of queries and that the number of such sessions is larger than or equal to the minimum model support count, then the contents of these sessions are immediately "large" and deserve the name of association combinations. Therefore, these sessions can skip the processing afterwards. For example, as Figure 6(b) shows, S_{10}, S_{11}, and S_{12} share identical query contents, and their number adds up to be 3, which is equal to the minimum model support count. Therefore, these three sessions are to be directly sorted out to establish the association combination {sql 2, sql 3, sql 4, sql 5, sql 6} without going through the processing afterwards.

[P2] Establish $same(S_i, S_j)$ for $S_j \in DM(S_i)|_{\hat{i}}, j \neq i$

In this step, S_i acts as the primary session, and the compared sessions S_j take turns to compare with S_i. After the comparisons, the same query codes sql are organized into $same(S_i, S_j)$. For example, in Figure 6, S_2 = (sql 1, sql 2, sql 3, sql 4, sql 5, sql 6, sql 7, sql 8), and S_1 = (sql 1, sql 2, sql 3, sql 4, sql 5, sql 6, sql 7, sql 8, sql 9). Therefore, $same(S_2, S_1)$ = { sql 1, sql 2, sql 3, sql 4, sql 5, sql 6, sql 7, sql 8}. Keep on with the work, and we can get the result in Figure 6(c): $same(S_2, S_j)$, where j = 1, 3, 4, 5.

[P3] Build up t complex sets com_same_{ij} for j = 1, 2, …, t, where $t = C_{f-1}^{|DM(S_i)|-1}$

Organizing the elements in any f-1 sets of the name $same(S_i, S_j)$ based on the primary session S_i, where the total number of such sets named $same(S_i, S_j)$ is $|DM(S_i)| - 1$, we can build up t (= $C_{f-1}^{|DM(S_i)|-1}$) sets of the kind com_same; that is, $com\_same_{ij} = \{S_i, same(S_i, S_{i1}), same(S_i, S_{i2}), \ldots, same(S_i, S_{i(f-1)})\}$, where j = 1, 2, …, $C_{f-1}^{|DM(S_i)|-1}$, and $same(S_i, S_{i1})$, $same(S_i, S_{i2})$, …. $same(S_i, S_{i(f-1)})$ are any f-1 elements in the set $same(S_i, S_j)$. For example, the primary session is S_2 in Figure 6(c). There are four sets of the name $same(S_2, S_j)$, where j = 1, 3, 4, 5, and the minimum model support count is 3. As a result, we can build up $C_{3-1}^{4} = 4! / (2!\ 2!) = 6$ sets of the kind com_same.

[P4] Derive large itemset R_{ij} from the sets com_same_{ij}

Examining the content of every com_same set, if the number of appearances of a same sql is equal to the minimum model support count, then we can put this sql in the large itemset. In our example, $com\_same_{21} = \{S_2, same(S_2, S_1), same(S_2, S_3)\}$. Here, the number of

Session code	Codes of queries made in each session
S_1	(sql 1, sql 2, sql 3, sql 4, sql 5, sql 6, sql 7, sql 8, sql 9)
S_2	(sql 1, sql 2, sql 3, sql 4, sql 5, sql 6, sql 7, sql 8)
S_3	(sql 1, sql 2, sql 4, sql 6, sql 7, sql 8)
S_4	(sql 1, sql 2, sql 4, sql 5, sql 6, sql 8)
S_5	(sql 1, sql 3, sql 4, sql 5, sql 6, sql 8)
S_6	(sql 1, sql 3, sql 5, sql 6, sql 8, sql 10, sql 12)
S_7	(sql 2, sql 6, sql 8, sql 10, sql 11)
S_8	(sql 3, sql 5, sql 6, sql 7, sql 11, sql 13)
S_9	(sql 4, sql 6, sql 8, sql 10, sql 12, sql 13)
S_{10}	(sql 2, sql 3, sql 4, sql 5, sql 6)
S_{11}	(sql 2, sql 3, sql 4, sql 5, sql 6)
S_{12}	(sql 2, sql 3, sql 4, sql 5, sql 6)

Fig.6(a) Query sessions

Session code	Codes of queries made in each session
S_1	(sql 1, sql 2, sql 3, sql 4, sql 5, sql 6, sql 7, sql 8, sql 9)
S_2	(sql 1, sql 2, sql 3, sql 4, sql 5, sql 6, sql 7, sql 8)
S_3	(sql 1, sql 2, sql 4, sql 6, sql 7, sql 8)
S_4	(sql 1, sql 2, sql 4, sql 5, sql 6, sql 8)
S_5	(sql 1, sql 3, sql 4, sql 5, sql 6, sql 8)

Note: S_{10}, S_{11}, and S_{12} are drawn out due to "Large."

Fig. 6(b) $MD(S_2)|_{\hat{i}=60\%}$

same(S_i, S_j)	Codes of queries
same(S_2, S_1)	{ sql 1, sql 2, sql 3, sql 4, sql 5, sql 6, sql 7, sql 8}
same(S_2, S_3)	{ sql 1, sql 2, sql 4, sql 6, sql 7, sql 8}
same(S_2, S_4)	{ sql 1, sql 2, sql 4, sql 5, sql 6, sql 8}
same(S_2, S_5)	{ sql 1, sql 3, sql 4, sql 5, sql 6, sql 8}

Fig. 6(c) same(S_2, S_j) for $MD(S_2)|_{\hat{i}=60\%}$

appearances is three for each of sql 1, sql 2, sql 4, sql 6, sql 7, and sql8, which is equal to the preset minimum model support count 3. Therefore, we can derive the association combination {sql 1, sql 2, sql 4, sql 6, sql 7, sql 8}. Combining the derived association combination sets by the rule "maximal," we can come to the final association combination set R_i.

The following examples are two cases for illustrating the Apriori-Model procedure:

Example 1

(1) Sort out sessions that are large from $DM(S_i)|_{\hat{i}}$ and build up same(S_i, S_j)

In Figure 6(a), the query sessions are made by many decision makers for their individual objectives; therefore, we can establish decision models following Steps 1 through 3. Suppose the minimum model affinity is set to be 60%, then we can set up two decision models $MD(S_2)|_{\hat{i}=60\%}$ = {S_1, S_2, S_3, S_4, S_5} and $MD(S_5)|_{\hat{i}=60\%}$ = {S_1, S_2, S_4, S_5, S_6}. Take $MD(S_2)|_{\hat{i}=60\%}$ for example, which is shown in Figure 6(b). Picking out sessions S_{10}, S_{11}, and S_{12} that are large, we get the association combination set {sql 2, sql 3, sql 4, sql 5, sql 6}. Then we derive same(S_2, S_j), where j = 1, 3, 4, 5 as Figure 6(c) shows.

(2) Build up complex set com_same

In case the minimum model support is 60%, then the minimum model support count is 4 ×60% 3. That is to say, this complex set com_same can be any set based on the primary session S_2 with any two other sets inside that are members of the family same(S_2, S_j) (j = 1, 3, 4, 5). In other words, such complex sets include com_same_{21} = { S_2, same(S_2, S_1), same(S_2, S_3)}, com_same_{22} = { S_2, same(S_2, S_1), same(S_2, S_4)}, com_same_{23} = { S_2, same(S_2, S_1), same(S_2, S_5)}, com_same_{24} = { S_2, same(S_2, S_3), same(S_2, S_4)}, com_same_{25} = { S_2, same(S_2, S_3), same(S_2, S_5)}, and com_same_{26} = { S_2, same(S_2, S_4), same(S_2, S_5)}.

(3) Derive the large itemset from the complex set com_same

In com_same_{21} = { S_2, same(S_2, S_1), same(S_2, S_3)}, sql 1, sql 2, sql 4, sql 6, sql 7, and sql 8 are query codes that appear three times each; in other words, they meet the requirement that the minimum model support count is equal to 3. Therefore, we can derive a second association combination set {sql 1, sql 2, sql 4, sql 6, sql 7, sql 8} here. Then, according to the complex set com_same_{ij} (i = 2, j = 2, 3, 4, 5, 6) built up just above, we can come up with association combination sets {sql 1, sql 2, sql 4, sql 5, sql 6, sql 8}, { sql 1, sql 3, sql 4, sql 5, sql 6, sql 8}, { sql 1, sql 2, sql 4, sql 6, sql 8}, { sql 1, sql 4, sql 6, sql 8}, and { sql 1, sql 4, sql 5, sql 6, sql 8}. Again, following the rule of "maximal," the large itemsets for $MD(S_2)|_{\hat{i}=60\%}$ are R_{21} = {sql 2, sql 3, sql 4, sql 5, sql 6}, R_{22} = { sql 1, sql 2, sql 4, sql 6, sql 7, sql 8}, R_{23} = { sql 1, sql 2, sql 4, sql 5, sql 6, sql 8}, and R_{24} = { sql 1, sql 3, sql 4, sql 5, sql 6, sql 8}, and $R_2 = R_{21} \cup R_{22} \cup R_{23} \cup R_{23}$. To put it another way, to come to the same result, our Apriori-Model needs to do 5 model support calculations on com_same_{ij} (i = 2, j = 1, 2, 3, 4, 5, 6), while Aprioriall [1] has to go all the way through 144 itemset model support calculations. In addition, according to the empirical studies in [1][4][5], model support calculations are the major factor that affects the efficiency of the practice of association models.

Example 2

(1) Sort out sessions that are large from $DM(S_i)|_{\hat{i}}$ and build up same(S_i, S_j)

Take $DM(S_2)|_{\hat{i}=50\%}$ = {S_1, S_2, S_4, S_5} built up through Steps 1 to 3 for example. We can establish the same(S_2, S_j) in Figure 6(d). In this example, there are no large sessions.

same(S_i, S_j)	Codes of queries
same(S_2, S_1)	{ sql 1, sql 3, sql 4}
same(S_2, S_4)	{ sql 1, sql 3, sql 5}
same(S_2, S_5)	{ sql 4, sql 5}

Fig. 6(d) same(S_2, S_j) for $MD(S_2)|_{\hat{i}=50\%}$

(2) Build up complex sets com_same

If the minimum model support is set to be 50%, then the minimum model support count is 3 ×50% 2. That means the complex set com_same can be any set based on the primary session S_2 with any extra set in the family of same(S_2, S_j) (j = 1, 4, 5). Such complex sets include com_same_{21} = { S_2, same(S_2, S_1)}, com_same_{22} = { S_2, same(S_2, S_4)}, and com_same_{23} = { S_2, same(S_2, S_5)}.

(3) Derive large itemsets from complex sets com_same

In com_same_{21} = { S_2, same(S_2, S_1)}, sql 1, sql 3, and

sql 4 are the query codes that appear twice and thus meet the requirement that the minimum model support

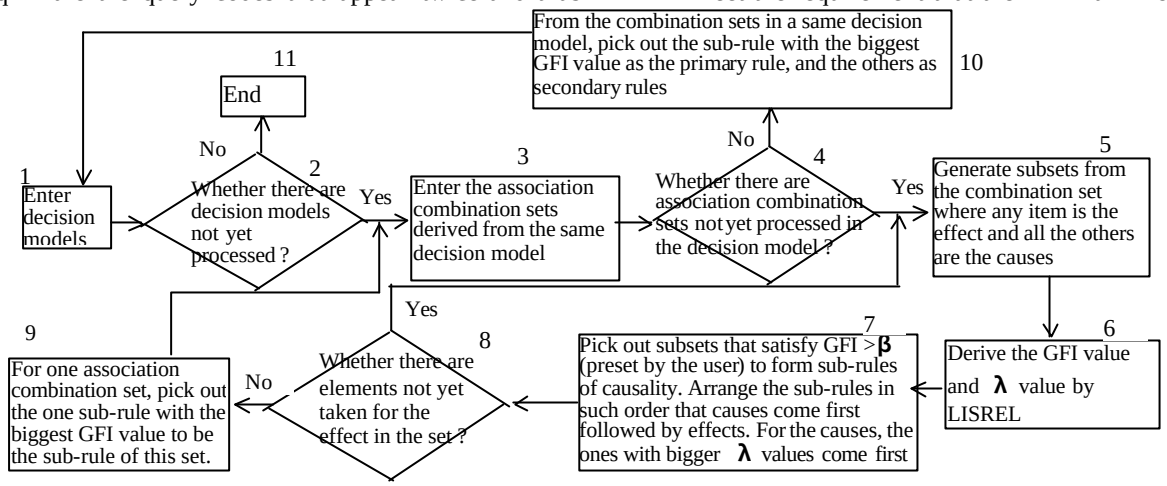


Fig. 7 Procedure of building up rules of cause-effect relationships by LISREL

count is equal to 2. Therefore, we come to the first association combination set { sql 1, sql 3, sql 4} here. From the complex sets com_same$_{ij}$ (i =2, j = 2, 3) derived just above, we can deduce two association combination sets { sql 1, sql 3, sql 5} and { sql 4, sql 5}. In this place, to get the same association combination sets, our Apriori_Model only needs 3 model support calculations on com_same$_{ij}$ (i = 2, j = 1, 2, 3), while Apriorial [1] has to do 16 itemset model support calculations.

Step 5 Establishing cause-and-effect association rules by LISREL

Figure 7 is the flowchart of the procedure to establish causality rules by the linear structure relation mode. First, the decision models built up in Step 1 are to be entered up. If there are no decision models not yet processed, then we come to an end here (see parts 1 and 2 of Figure 7). Then, we access the association combination sets not yet processed in the same decision model (see parts 3 and 4 of Figure 7). In Step 4, Example 2, we deal with association combinations by means of Apriori-Model, and three association combination sets are generated from MD(S_2)| î=50%, where one of the association combination sets is {sql 1, sql 3, sql 5}. In this very association combination set, if sql 1 stands for the query code for "salary imbalance index," sql 3 stands for the query code for the " income imbalance index," and sql 5 stands for the query code for the "average number of crimes committed," then we can learn from part 5 of Figure 7 that, when deriving association rules of causal relationships, we need to pick out every element in the rules as the effect and the other elements as causes to discuss the appropriateness of the cause-effect model. For example, if we pick out sql 3 as the effect, then we can come to the sub-rule sql 1, sql 5->sql 3. The linear structure relation model can be applied as Figure 8 (or parts 6 and 7 of Figure 7) shows. The test values concerned are listed in Figure 9. According to the test result, the GFI value is 0.702, which is greater than the preset GFI threshold (supposedly 0.6). In addition, the p-value for every element is less than 0.05, and the regression coefficients are respectively 1 and 0.657. As a result, we get one association sub-rule of causality sql 1, sql 5->sql 3. Of course, the remaining sub-rules sql 1, sql 3->sql 5 and sql 3, sql 5->sql 1 should be further processed (by repeating parts 5, 6, and 7 of Figure 7). When this

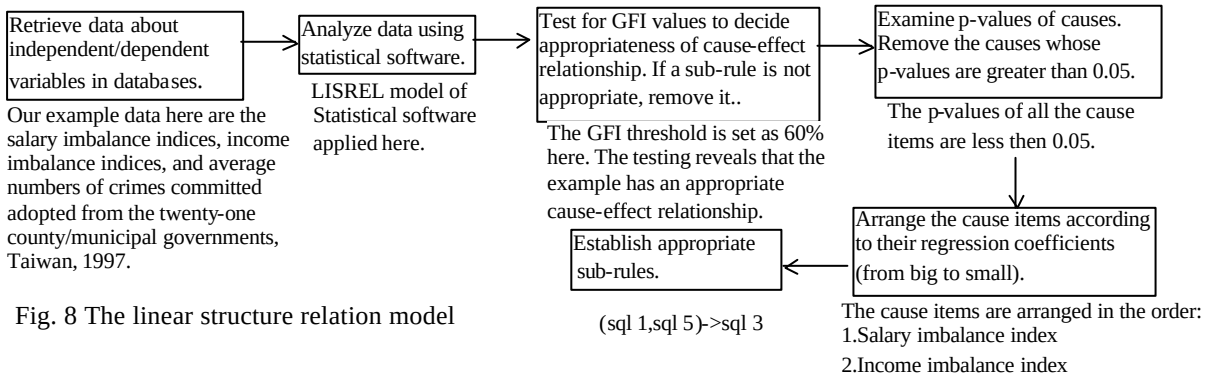


Fig. 8 The linear structure relation model

Overall Model Fit

Statistic	Value	P-value
Chi-square	56.6461	0.01
GFI	0.702	
AGFI	0.504	
RMR	0.15	

Measurement Model Result

Parameter	Estimate	P-value
Y1	1*	
X1	1*	
X2	0.659	0.000

Notes:
1. X1: salary imbalance index, X2: income imbalance index, and Y1: average number of crimes committed.
2. The symbol * means the value is kept constant for other variables to compare with.

Fig. 9 Test values of relation models

association combination set is done and yet there are still other association combination sets unprocessed in the same model such as {sql 1, sql 3, sql 4} and {sql 4, sql 5}, we must also repeat the steps in parts 3 through 7 of Figure 7 on them.

Please note here that there may be more than one association combination in a decision model. For example, in $MD(S_2)|_{\hat{\imath}=50\%}$, there are as many as three association combination sets {sql 1, sql 3, sql 4}, {sql 1, sql 3, sql 5}, and {sql 4, sql 5}. Because there is only one single objective to one decision model, from $MD(S_2)|_{\hat{\imath}=50\%}$, no matter how many association sub-rules of causality we can derive via the linear structure relation model, only the sub-rule with the highest GFI value is chosen. The reason is that, among the sub-rules derived from one model, the one with the highest GFI value is the one with the strongest cause-effect relationship and thus is the one we need.

The association rules are picked out the way as follows. In {sql 1, sql 3, sql 5} from $MD(S_2)|_{\hat{\imath}=50\%}$, we get the following two association sub-rules of causality sql 1, sql 5->sql 3 (GFI=0.702) and sql 3, sql 5->sql 1 (GFI=0.601). Out of the two, the sub-rule sql 1, sql 5->sql 3 has a bigger GFI value, which means the cause-effect relationship indicated by this sub-rule is stronger; therefore, we take it (see part 9 of Figure 7). On the other hand, $MD(S_2)|_{\hat{\imath}=50\%}$ still has one other association combination set {sql 1, sql 3, sql 4}, from which we get three association sub-rules of causality sql 1, sql 4->sql 3 (GFI=0.500), sql 3, sql 1->sql 4 (GFI=0.503), as well as sql 3, sql 4->sql 1 (GFI=0.402). Out of the three, sql 3, sql 1->sql 4 has the biggest GFI value, so the cause-effect relationship it indicates must be the strongest. Therefore, we take sql 3, sql 1->sql 4. After we have gone through all the association combination sets, we compare the GFI values of the sub-rules picked out. Among them, the sub-rule with the biggest GFI value (such as sql 1, sql 5->sql 3 in our example) is taken for the primary rule, and the others (such as sql 3, sql 1->sql 4) will serve as secondary rules (refer to part 10 of Figure 7). Since the GFI value indicates the probability of the existence of a cause-effect relationship, we should of course reserve rules with high GFI values for later use in the pre-fetching of materialized views as well as the intelligent query paths.

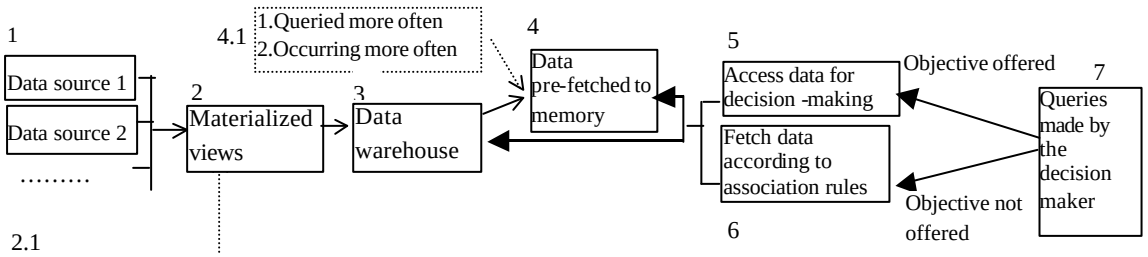
5. EVALUATIONA OF IMVIP MECHANISM

The raw data we adopt in this paper for the empirical analysis of our new method are the queries made in Taiwan government database warehouse within a certain period of time. Based on the raw data, we derive six association rules of causality as shown in part 2.1 of Figure 10, which will help us make it through the inferences in the rest of this section. Figure 10 below illustrates how our IMVIP mechanism works. When a decision maker proceeds with queries for data, the cause-and-effect association rules can help with the data warehouse operation in the following steps:

Step 1. The data warehouse (see part 3 of Figure 10) can fetch and organize the associated data in advance that are originally shattered in different databases (see part 1 of Figure 10). However, if there should be a big gap between the pre-fetched data and the data really practically needed, the resources put into the construction of the data warehouse would be a waste. In this paper, the cause-and-effect association rules derived to build up the materialized views (see part 2 of Figure 10) are directed at the needs of the decision makers (see part 2.1 of Figure 10) in order to enhance the efficiency of the data warehouse.

Step 2. When offering data in the data warehouse for use, we can fetch the data within the association rules whose numbers of queries made by the decision maker are higher (for example, in rule 2 of part 2.1 in Figure 10, higher numbers of queries are made for "number of couples married," "population," "sex ratio," "income distribution," and "population growth") as well as the data that show up more often in the derived rules (such as "population," which appears three times in the six rules) and store them in memory in advance when the data warehouse is being setup (see parts 4 and 4.1 in Figure 10). This way, we can save the time that would otherwise be wasted on storing media and searching for data in the hard disk. Of course, the threshold here needs to be adjusted according to the size of the computer memory used as well as the sizes of the databases.

Step 3. When searching for data (part 7 of Figure 10), if the decision maker offers the objective (for example, to infer the population growth ratio), then the system can draw out the associated data (namely the "number of couples married," "population," "sex ratio," "income distribution," and "population growth") (see part 5 of Figure 10) according to rule 2 in part 2.1 of Figure 10. However, if the decision maker does not reveal the objective, then thing will go another way. Suppose the first query the decision



2.1

Rule No.	Rule description
1	salary imbalance index, income imbalance index → number of crimes committed
2	number of couples married, population, sex ratio, income distribution → population growth
3	number of companies, number of lands traded, population, unemployment rate → growth of local tax
4	degree of urbanization, education, age group, population, sex ratio → number of social worker needed
5	education, degree of urbanization, income distribution, street length, growth of motor vehicles → number of traffic accidents
6	number of cars increased, growth of public transportation, income distribution, population → number of parking spaces needed

Fig. 10 Illustration of the IMVIP mechanism

maker ma[illegible]opulation. Then, the system will show the d[illegible]d with population (for example, "number o[illegible]narried," "sex ratio," "income distribution," and "population growth" in rule 2; "number of companies," "number of lands traded," "unemployment rate," a[illegible]h of local tax" in rule 3; "degree of urbanization," "education," "age group," "sex ratio," and "number of social workers needed" in rule 4; as well as "number of cars increased," "growth of public transportation," "income distribution," and "number of parking spaces needed" in rule 6) for the decision maker to choose from for further queries. Then, if the decision maker makes a second query for sex ratio, the range of associated data will be narrowed down to rules 2 and 4. After that, if the decision maker makes a third query for number of couples married, then the decision maker's objective is predicted to be pointing at rule 2, and the system will offer related data as such (see part 6 in Figure 10). Thus, the system can provide the decision maker with intelligent data pre-fetching along the decision-making path, which will dramatically enhance the efficiency for the decision-making process.

Assumptions are made for the six association rules in part 2.1 of Figure 10 : (1) The total number of the items is 22 (where items that different rules have in common, such as "population" that appears in rules 2, 3, 4, and 6, are counted only once). (2) 11 queries are applied to rules 1, 2, and 3, individually. (taking up 10.18% of the total number of queries) 25 queries are applied to rules 4, 5, and 6, individually. (taking up 23.15% of the total number of queries).

Table 1 is a list of six different methods of data warehouse construction codenamed A through F. Among them, none of methods A through D uses the system proposed in this paper. On the contrary, both methods E and F are followers of our IMVIP mechanism. On the other hand, rows a through f are for evaluation items for those data warehouse construction methods to examine the working efficiency performances as to queries for decision-making. For example, "a: number of data items fetched from data sources (4)" is to put down the number of data items queried that cannot but be fetched from their very sources outside the data warehouse, and the number 4 inside the parentheses indicates the time complexity for fetching one data item. If the query a certain decision maker makes is for rules 3 and 4 in part 2.1 in Figure 10, then Cell A-a will show 50%, indicating that only half of the data needed can be found in the data warehouse with the other half outside in their individual sources. There are totally 10 data items in rules 3 and 4, and therefore there are still 5 other data items to be fetched outside the data warehouse. Cell A-b is for the summary data acquired by calculating and organizing other data. In rules 3 and 4, such data include "degree of urbanization," "age group," and "sex ratio." Cells A-c and A-d suggest that the data warehouse offers no intelligent decision-making path, and thus the number of data items shown on the screen is 22, whether or not the objective has been provided. Cell A-e reveals that 10 data items must be drawn out from the hard disk. Finally, the number (113=4×5+3×3+1×22+1×22+4×10) in Cell A-f is figured out by multiplying all the time complexity indices in the cells above. It is used to indicate the time complexity of the particular data warehouse construction method. By the same token, we can get the corresponding information as to different data warehouse construction methods in Columns B through D in Table 1.

As for data warehouse construction methods E and F, because IMVIP mechanism is employed, there is already no need to search individual databases for data, nor is there any need for data calculation and integration. Therefore, Cells E-a, E-b, F-a, and F-b are all 0's. Cells E-c and F-c show that the decision maker faces the 10 data items in rules 3 and 4 when she/he has revealed the objective. Then, Cells E-d and F-d indicate that it takes two steps for our new mechanism to reach the data in rules 3 and 4 when the decision maker's objective remains unknown. Here, the first step is to fetch the data in rule 3. If the decision maker chooses population, then she/he is going to face the 17 data items in rules 2, 3, 4, and 6 (excluding population of course). If the second choice is unemployment rate, then

Table 1. Evaluation of data warehouse construction methods

Methods of data warehouse construction / Evaluation items	Proposed mechanism not employed				Proposed mechanism employed	
	A: 50% of the data needed can be found in the data warehouse. summary data not processed	**B**: 50% of the data needed can be found in the data warehouse. summary data processed	**C**: 90% of the data needed can be found in the data warehouse. summary data not processed	**D**: 90% of the data needed can be found in the data warehouse. summary data processed	**E**: 100% of the data needed can be found in the data warehouse. Summary data processed. Data items put in memory whose query is applied more than 20% of the total number of queries	**F**: 100% of the data needed can be found in the data warehouse. Summary data processed. Data items put in memory whose query is applied more than 10% of the total number of queries
a: number of data items fetched from data sources (4)	5	5	1	1	0	0
b: number of summary data items processed (3)	3	0	3	0	0	0
c: number of data items offered when user has revealed the objective (1)	22	22	22	22	10	10
d: number of data items offered when user has not revealed the objective (1)	22	22	22	22	17	17
e: number of data items to be drawn out from hard disk when queried (4)	10	10	10	10	4	0
f: total time complexity	113	104	97	88	43	27

the decision maker is going to face the remaining 3 data items in rule 3. Now, here comes the second step, which is for the access of the data in rule 4. If the decision maker's first choice is population, then, again, she/he is going to face the 17 data items in rules 2, 3, 4, and 6 except for population itself. After that, if the second choice is education, then she/he will face the 4 remaining items in rule 4. This way, although the number of data items faced varies from choice to choice, the total number is always 17. As for Cell E-e, because the data whose query is applied more than 20% of the total number of queries have already been put in memory, we have got all the items in rule 4 readily available with the four items "number of companies," "number of lands traded," "unemployment rate," and "growth of local tax" not yet loaded in. Then, as Cell F-e reveals, when the data whose query is applied more than 10% of the total number of queries have already been put in memory, we have got all the items in rules 3 and 4 ready for query in memory. Finally, the time complexity indices in Cells E-f and F-f can also be figured out according to the above cells. According to the total time complexity in Cells E-f and F-f, we can find that the performance is always better if the IMVIP mechanism is employed.

From the evaluation above, we find out there are three key factors that affect the efficiency of data warehouse operations, and they are: whether or not the data items stored can meet the need for decision-making, whether or not the system can offer intelligent query paths, as well as whether or not the data items needed can be pre-fetched and kept in memory. The time complexity indices for methods of data warehouse construction in Table 1 have demonstrated that our intelligent materialized views pre-fetching mechanism, performing quite well in all the three key factors above, can offer appropriate problem-solving plans and will be empirically helpful for data warehouse construction.

6. CONCLUSIONS

The database components of a data warehouse keep in store the basic information that decision makers need when they have to make decisions of all kinds. Since the data sources usually locate separately in a wide variety of application systems, where even the working platforms are very probably heterogeneous, if we can build up definitions for materialized views that are able to live up to the needs of decision makers, then we can be successful in lowering the probability of cases where the data needed have to be searched for outside of the data warehouse. As a result, we will be able to not only reduce the load on computer resources but also enhance the efficiency in decision-making.

To satisfy the requirement above, in this paper, we have proposed an IMVIP mechanism to derive materialized views fitting various decision makers' needs from the Apriori-Model and the linear structure relation model. Exploring every single query that each decision maker makes for a certain data item, this intelligent system can derive the association rules as to the most-often-queried data items for every possible query objective. Through the inference of the association rules among queries, in the data management system, the data about to be queried by the decision maker can thus be pre-fetched. As a result, the data access time can be shortened, and the decision-making efficiency improved. In addition, by means of the linear structure relation model, the system can determine the cause-effect relationships among the queried data items in the association rules. These cause-and-effect association rules can further serve as more-pertinently-to-the-point references for decision makers. As a result, such rules will be great help with the construction and performance of data warehouses.

REFERENCES

[1] Agrawal, Rakesh & Srikant, Ramakrishnan, "Mining Sequential Patterns," *Data Engineering*, 1995, 3-14.

[2] Benamati, John & Lederer, Albert L., "An Empirical Study IT Management and Rapid IT Change," *Proceedings of the 1999 ACM SIGCPR Conference on Computer Personnel Research*, 1999, 144-153.

[3] Berson, Alex & Smith, Stephen J., *Data Warehousing, Data Mining, & OLAP*, McGraw-Hill Companies, 1997.

[4] Cai, C. H., Fu, Ada W. C., Cheng, C. H. & Kwong, W. W., "Mining Association Rules with Weighted Items," *Proceedings of Database Engineering and Applications Symposium*, 1998, 68-79.

[5] Han, Jiawei & Fu, Yongjian, "Mining Multiple-Level Association Rules in Large Databases," *IEEE Transactions on Knowledge and Data Engineering*, 1999,11(5), 798-805.

[6] Hoel, Paul G., Port, Sidney C. & Stone, Charles J., *Introduction to Probability Theory*, Houghton Mifflin Company, 1985.

[7] Johnson, Dallas E., *Applied Multivariate Methods for Data Analysts*, Brooks/Cole Publishing Company, 1998.

[8] Kawano, Hiroyuki & Hasegawa, Toshiharu, "Mondou: Interface with Text Data Mining for Web Search Engine, "*Proceedings of 31th Annual Hawaii International Conference on System Sciences*, 1998, 275-283.

[9] Krippendorf, Micheal & Song, Il-Yeol, "The Translation of STAR Schema into Entity-Relationship Diagrams," *Database and Expert Systems Applications*, 1997, 390-395.

[10] Liang, Weifa, Li, Hui, Wang, Hui & Orlowska, Maria E., "Making Multiple Views Self-Maintainable in a Data Warehouse," *Data & Knowledge Engineering*, 1999,30, 121-134.

[11] Ragel, A. & Cremilleux, B., "MVC - A Preprocessing Method to Deal With Missing Values," *Knowledge-Based Systems*, 1999,12, 285-291.

[12] Samtani, S. & Kumar, V., "Maintaining Consistency in Partially Self-Maintainable Views at the Data Warehouse," *Database and Expert System Applications*, 1998, 206-211.

[13] Theodoratos, Dimitri & Sellis, Timos, "Designing Data Warehouses," *Data & Knowledge Engineering*, 1999,31, 279-301.