3. Problem definition

In this section, we first define a *multi-dimension transaction database* **MD**, to differentiate it between the transaction databases **D** in traditional association rules. Then, we discuss the concept hierarchy for each dimension in **MD**. We also develop a mechanism to produce *multi-dimension patterns* with different granularities related to each dimension in this section. At last, we define the *multi-dimension association rules* in this paper.

3.1. Multi-dimension transaction database

After joining several relative tables, we can obtain a big table to present not only which items are bought in a transaction, but also when, where and who performed this transaction. An example of this kind of table is shown in Fig. 3.1. We assume a transaction database with other attributes as a *multi-dimension transaction database* **MD**, to differentiate it between transaction databases **D**.

T_ID	Date	Branch_No	Occupation	Sex	Age	Transaction content
001	05/03/01	003	Student	F	23	Bread, Milk
002	05/03/03	003	Student	M	14	Candy, Juice
003	05/03/01	003	Doctor	M	47	Beer, Diaper
004	05/03/01	003	Teacher	F	34	Juice, Tomato, Orange

Figure 3.1 Multi-dimension transaction database MD

In above table, we have five addition attributes: “Date” and “Branch_NO” are the time and place this transaction submitted, “Occupation”, “Sex” and “Age” are the profile of customer in this transaction. Except “Transaction content”, we assume each non-key attributes is a dimension in this paper, and each record in above table is a transaction. According to this definition, a transaction in **MD** is a tuple with the form in Fig. 3.2, where “T_ID”, transaction id, is used to identity each transaction, “Value

of *Dimension*$_x$" is the value of the x^{th} dimension in this transaction, and "TC", transaction content, is a set of items which are bought in this transaction.

T_ID	Value of *Dimension*$_1$	Value of *Dimension*$_2$	…	Value of *Dimension*$_n$	TC

Figure 3.2 A transaction in MD

3.2. A dimension in MD

A dimension in **MD** is a non-key attribute except "Transaction content", and we use dim_x to denote the x^{th} dimension in **MD**. To produce multi-dimension patterns at varying levels of abstraction in each dimension, we use a concept hierarchy tree to represent a "specific-to-general" way for each dimension. The $\mathbf{CH_x}$ is the concept hierarchy of dim_x. An example of $\mathbf{CH_x}$ is shown in Fig. 3.3. A dimension concept hierarchy $\mathbf{CH_x}$ is a tree structure with a root "any" covering all nodes in $\mathbf{CH_x}$. An ancestor node in $\mathbf{CH_x}$ covers all its descendant nodes in $\mathbf{CH_x}$. Each node in $\mathbf{CH_x}$ is a possible value of different granularities in dim_x. The root "any" is the biggest granularity of every dimension. The leaves of $\mathbf{CH_x}$ are *dimension atoms* of $\mathbf{CH_x}$, and other non-leaf nodes in $\mathbf{CH_x}$ are *dimension compounds* as shown in Fig. 3.3. The *dimension atoms* and *dimension compounds* will be discussed respectively later.

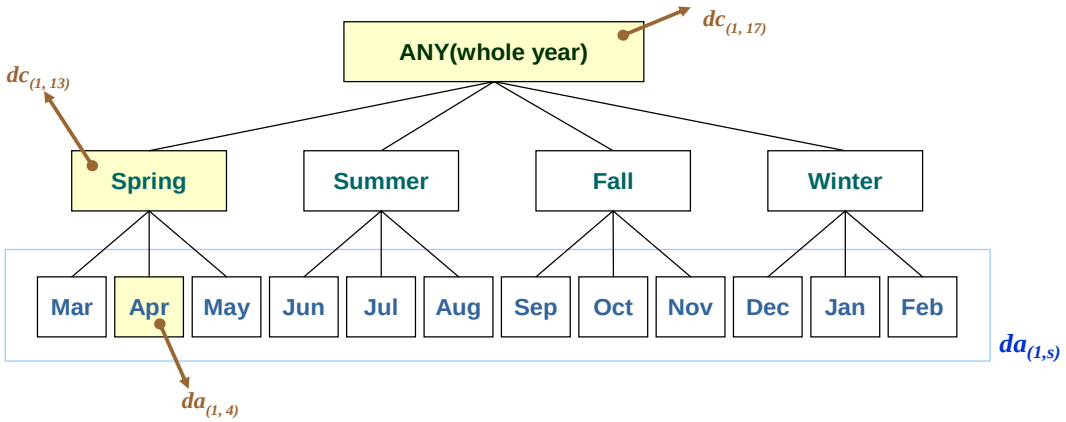

Figure 3.3 CH_1 for dimension "Date"

Definition 1. dim_x, the x^{th} dimension of **MD**, is a non-key attribute except "Transaction content", and $\mathbf{CH_x}$ is the concept hierarchy of dim_x.

3.2.1. Dimension atom

The leaves of $\mathbf{CH_x}$ are dimension atoms of $\mathbf{CH_x}$, since the other non-leaf nodes are compositions of them. We use $da_{(x,s)}$ to denote the s^{th} dimension atom of $\mathbf{CH_x}$. A dimension atom is a possible value of dim_x in **MD** or a grouping values for dim_x. For example, in the dimension "date", the granularity "a day" may be too trivial to make strategies. We use "a month" to be the granularity of *dimension atoms* in the concept hierarchy of this dimension. In the "Age" dimension, we also can discretize the numerical data into category one (for example: child, teens, youth, adult) to be dimension atoms. All the *dimension atoms* in the same $\mathbf{CH_x}$ are mutual disjoint and form a complete partition of whole range of this dimension. That is, all the dimension atoms in a dimension can be used to divide **MD** into several mutually disjoint slices, there is an example shown in Fig. 3.4. Suppose the whole range of the dimension "date" is a year "2005", and the granularity of the *dimension atoms* in this dimension is "a month", all the $da_{(1,s)}$ divide **MD** into twelve mutually disjoint slices, and the twelve slices form a complete partition of **MD**.

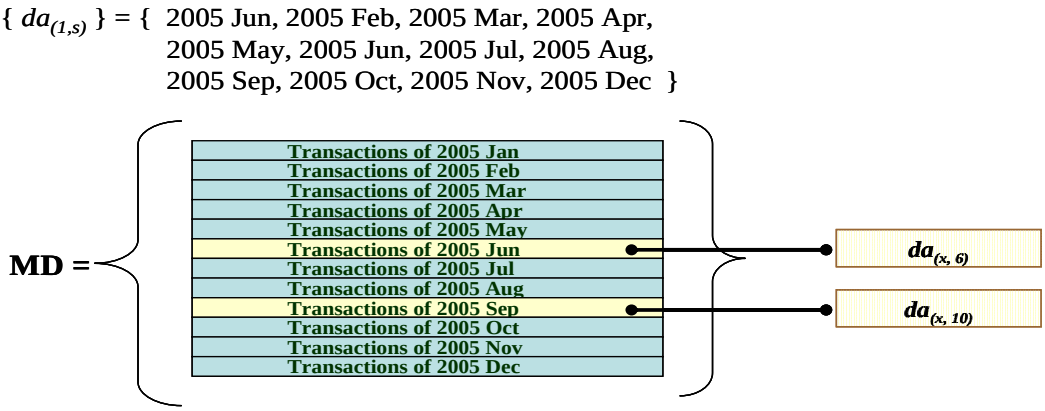

Figure 3.4 Example of dimension atom in dimension "Date"

Definition 2.1. $dim_x = \{ da_{(x, s)} \} = \{ da_{(x,1)}, da_{(x,2)}, \ldots, da_{(x, n)} \}$,

where $da_{(x, s)}$ (s = 1 to n) is the s^{th} *dimension atom* of $\mathbf{CH_x}$.

Definition 2.2. In the same dimension dim_x,

{transactions of $da_{(x,s1)}$} $\bigcap$ {transactions of $da_{(x,s2)}$} = ϕ (if s1 $\neq$ s2),

$\bigcup_s$ {transactions of $da_{(x, s)}$} = **MD**

3.2.2. Dimension compound

Other non-leaf nodes in $\mathbf{CH_x}$ are *dimension compounds*, since they are the combination of several *dimension atoms*. We use $dc_{(x,t)}$ to denote the t^{th} *dimension compound* in $\mathbf{CH_x}$. the *dimension compounds* in the same level are mutually disjoint. An example of *dimension compounds* is shown in Fig, 3.5. In Fig. 3.5, *dimension compound* "2005 Spring" is the combination of tree *dimension atoms*: "2005 Mar", "2005 Apr" and "2005 May".

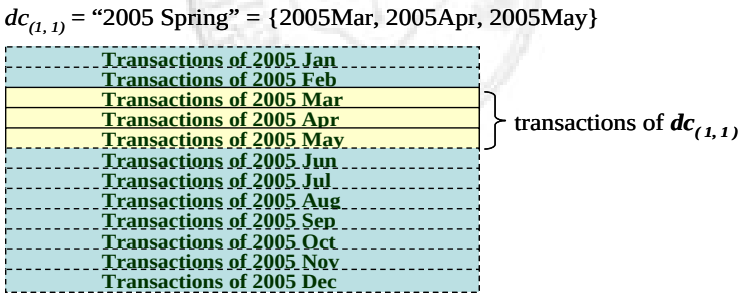

Figure 3.5 An example of dimension compound in dimension "Date"

Definition 3.1. $dc_{(x, t)}$, the t^{th} *dimension compound* in $\mathbf{CH_x}$, consists of one or more *dimension atoms* in dim_x, is a non-leaf node in $\mathbf{CH_x}$.

Definition 3.2. $dc_{(x, t)} = \{da_{(x, s1)}, da_{(x, s2)}, \ldots\}$

$|dc_{(x, t)}|$ is the number of *dimension atoms* which belong to $dc_{(x, t)}$,

$dc_{(x, t1)} \subseteq dc_{(x,t2)}$ if and only if $|dc_{(x,t1)}| \leq |dc_{(x,t2)}|$ and all *dimension atoms* which belong to $dc_{(x,t1)}$ also belong to $dc_{(x,t2)}$,

$dc_{(x,t1)} \bigcap dc_{(x,t2)} = \phi$, expect ($dc_{(x, t1)} \subseteq dc_{(x,t2)}$) or ($dc_{(x,t2)} \subseteq dc_{(x,t1)}$).

3.2.3. Concept hierarchy with lattice structure

To make our definition easier to understand, we define our concept hierarchy a tree structure. Actually, we also can define a lattice represented to improve the presenting ability of our concept hierarchy. An example in dimension "location" is shown in Fig. 3.6. In this concept hierarchy with a lattice structure, a successor node can has more than one predecessor. That is, the disjoint limit among *dimension compounds* in above definition can be ignored. Other definitions of a tree structure concept hierarchy above are also work in a lattice structure one in our work.

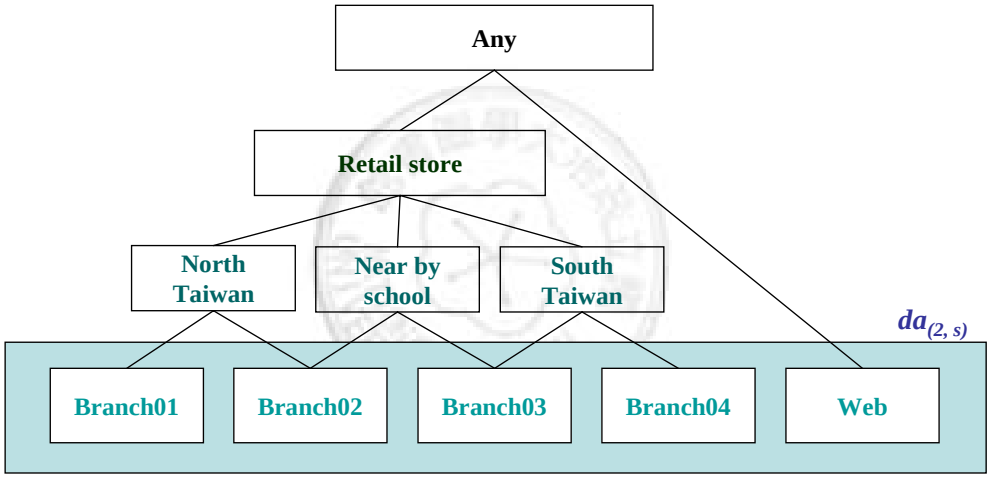

Figure 3.6 An example of lattice structure concept hierarchy

3.3. Multi-dimension pattern

Given a set of concept hierarchies {$\mathbf{CH_1}$, $\mathbf{CH_2}$, $\mathbf{CH_3}$, …, $\mathbf{CH_n}$}, where n is the number of dimensions. A *multi-dimension pattern P* has this form <p_1, p_2, …, p_n> together with a valid constraint. We use p_k to denote a pattern item related to dim_k, and p_k is a node in $\mathbf{CH_k}$. For example, <2005 Jan, North Taiwan, Student, Male, Child> and <2004 spring, Branch003, Any, Any, Any> are two *multi-dimension patterns*. For simplifying the representation of *multi-dimension patterns*, the pattern item "Any" can be ignored, since "Any" can be interpreted as "don't care". For example, <2004

spring, Branch003, Any, Any, Any> can be simplified to <2004 spring, Branch003>. If all of the *pattern items* in *P* are "Any", we call this pattern a *universal pattern*, and donated by P_u. The constraint valid is a Boolean function, this constraint serves two purposes. The first is to exclude the combinations that do not exit. For example, suppose we have a *multi-dimension pattern* <2004, Branch004>, but the retail store "Branch004" didn't open in 2004. The second purpose of the valid constraint is to exclude the patterns that we are not interested in. A *multi-dimension pattern* P_1 is covered by a *multi-dimension pattern* P_2, if and only if each pattern item p_{k1} in P_1 is covered by p_{k2} in P_2. For example, the <2005 Mar, North Taiwan, Student, Male, Child> is covered by <2005 spring, Taiwan, Any, Male, Any>.

Definition 4.1. *multi-dimension pattern* $P = <p_1, p_2, \ldots, p_n>$ together with a valid constraint, where pattern item p_k (k = 1 to n) is a node in $\mathbf{CH_k}$.

Definition 4.2. $|p_k|$ = the number of *dimension atoms* covered by p_k. If p_k is a *dimension atom*, $|p_k| = 1$.

Definition 4.3. Suppose P_1 and P_2 are two *multi-dimension patterns*,

$P_1 = <p_{11}, p_{12}, \ldots, p_{1n}> \subseteq P_2 = <p_{21}, p_{22}, \ldots, p_{2n}>$, if and only if

$\forall k$ (k = 1 to n) $p_{1k} \subseteq p_{2k}$.

Definition 4.4. $P = <p_1, p_2, \ldots, p_n>$ is a *universal pattern* and is denoted by P_u if and only if $\forall k$ (k = 1 to n) p_k = "Any".

3.3.1. Element patterns and generalized patterns

A *multi-dimension pattern* is an *element pattern* if and only if every p_k in this pattern is a *dimension atom*. On the other hand, if at least one of them is a *dimension compound*, we call this pattern a *generalized pattern*. For example, <2004 Mar, Branch003, Student, Female, Youth> is an *element pattern*, <2004 spring, Any, Any,

Female, Youth> is a *generalized pattern*, and both them are *multi-dimension patterns*. We use E_i to denote the i^{th} *element pattern*, and use G_j to denote the j^{th} *generalized pattern*.

Definition 5.1. A *multi-dimension pattern* $<p_1, p_2, \ldots, p_n>$ is an *element pattern* E_i if and only if $\forall k$ (k = 1 to n) p_k is a *dimension atom*.

Definition 5.2. A *multi-dimension pattern* $<p_1, p_2, \ldots, p_n>$ is a *generalized pattern* G_j if and only if $\exists k$ (k = 1 to n) p_k is a *dimension compound*.

3.3.2. Element segmentations

We can use *dimension atoms* of each dimension to dice the *multi-dimension transaction database* **MD** into several mutually disjoint hyper-cubes as in Fig. 3.7. Each hyper-cube enclosed by an *element pattern* E_i contains a set of transactions whose attribute of each dimension is covered by correspond p_k in E_i. We denote the set of transactions which are in the hyper-cube enclosed by an *element pattern* E_i as $T[E_i]$. We call $T[E_i]$ an *element segmentation* in **MD**. Given a minimum number of transactions *mintrans*, an *element segmentation* $T[E_i]$ is valid if and only if E_i is a valid *element pattern*, and the number of transactions in $T[E_i]$ exceed *mintrans*. An example of *element segmentation* which is enclosed by <2004 Apr, Branch003, Adult> is shown in Fig. 3.8.

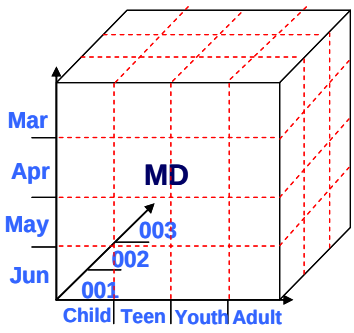

Figure 3.7 Hyper-cubes diced by dimension atoms.

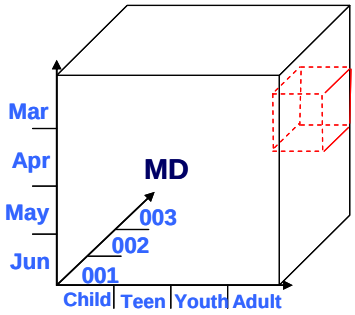

Figure 3.8 An example of element segmentation.

Definition 6.1. $T[E_i]$ = {transactions which belong to E_i},

$|T[E_i]|$ = the number of transactions which belong to E_i.

$T[E_i]$ is valid if and only if E_i is valid and $|T[E_i]| \geq mintrans$.

Definition 6.2. $T[E_{i1}] \bigcap T[E_{i2}] = \phi$ (if $i1 \neq i2$).

3.3.3. Combination segmentations

A generalized pattern also forms a hyper-cube in **MD**. We denote the set of transactions whose attribute of each dimension is covered by correspond p_k in generalized pattern G_j as $T[G_j]$. We call $T[G_j]$ an *combination segmentation* in **MD**, since a *combination segmentation* is composed of several *element segmentations*. Such conception is shown in Fig. 3.9. In Fig. 3.9, the *combination segmentation* <2004 spring, Branch003, Adult> is composed of three *element segmentations*: <2004 Mar, Branch003, Adult>, <2004 Apr, Branch003, Adult>, and <2004 May, Branch003, Adult>. Given a minimum number of *element segmentations* covered *mincover*, a *combination segmentation* $T[G_j]$ is valid if and only if G_j is a valid *generalized pattern*, and the number of valid *element segmentations* covered by $T[G_j]$ exceed *mincover*.

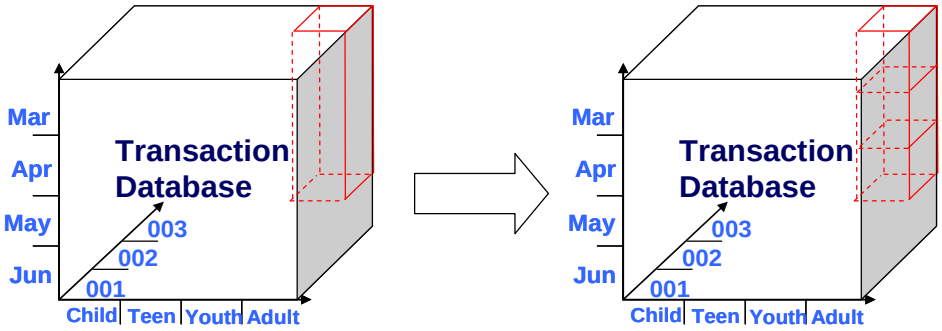

Figure 3.9 A combination segmentation is composed of element segmentations.

Definition 7. $T[G_j]$ = {transactions which belong to G_j},

$T[G_j]$ covers $T[E_i]$ if and only if $\forall$ transactions belong to $T[E_j]$ also belong to $T[G_j]$.

$|T[G_j]|$ = the number of valid $T[E_j]$ covered by $T[G_j]$.

$T[G_j]$ is valid if and only if G_j is valid and $|T[G_j]| \geq mincover$.

Lemma 1. Given a generalized pattern $G_j = <p_1, p_2, \ldots, p_n>$, the number of (valid and invalid) $T[E_i]$ covered by $T[G_j] = \prod_k(|p_k|)$.

3.4. Multi-dimension association rules

A multi-dimension association rule over a set of concept hierarchies is a pair (P, r), where P is a multi-dimension pattern and r is an association rule. In the following, we identity several classes of multi-dimension association rules on which we will focus in this paper.

3.4.1. Multi-dimension association rule w.r.t full match

Given a multi-dimension transaction database **MD**, a *minsup*, and a *minconf*. A *multi-dimension association rule w.r.t full match* (G_j, r) holds in **MD** if and only if the association rule r holds in every *element segmentation* $T[E_i]$ which are covered by $T[G_j]$. We draw on "full match" after this kind of multi-dimension association rules, since this kind of association rules not only hold in $T[G_j]$(Lemma 2), but also hold in

every meaningful dice of $T[G_j]$(Lemma 3).

Definition 8. Given a *multi-dimension transaction database* **MD**, a *minsup*, and a *minconf*. (G_j, r) is a *multi-dimension association rule w.r.t full match* if and only if G_j is valid and $\forall\, T[E_i] \subset T[G_j]$, r holds in $T[E_i]$ or $T[E_i]$ is invalid.

Lemma 2. If (G_j, r) is a *multi-dimension association rule w.r.t full match* in **MD,** r must be an association rule in $T[G_j]$. (A prove is given in Appendix B)

Lemma 3. If (G_j, r) is a *multi-dimension association rule w.r.t full match* in **MD**, and G_j is an universal pattern P_u, which means that $T[G_j]$ = **MD**, r must be an association rule in any dice of **MD**. (A prove is given in Appendix B)

3.4.2. Multi-dimension association rule w.r.t relaxed match

Multi-dimension association rule w.r.t full match may be too restrictive in some application. Instead, we may only require that the association rule holds in "enough" number of *element segmentations*. Given a multi-dimension transaction database **MD**, a *minsup*, a *minconf*, and a match ratio m ($0<m\leq 1$), a *multi-dimension association rule w.r.t relaxed match* (G_j, r) holds in **MD** if and only if at least $100m$% of the *element segmentation* $T[E_i]$ covered by $T[G_j]$, the association rule r holds in $T[E_i]$.

Definition 9. Given a *multi-dimension transaction database* **MD**, a *minsup*, a *minconf*, and a *match ratio* m ($0<m\leq 1$), (G_j, r) is a *multi-dimension association rule w.r.t relaxed match* if and only if G_j is valid and (number of valid $T[E_i]$, where $T[E_i] \subset T[G_j]$ and r holds in $T[E_i]$) / (number of valid $T[E_i]$, where $T[E_i] \subset T[G_j]$) $\geq m$.