Distributed Closed Pattern Mining in Multi-Relational Data based on Iceberg Query Lattices: Some Preliminary Results

Hirohisa Seki* and Sho-ich Tanimoto

Dept. of Computer Science, Nagoya Inst. of Technology,
Showa-ku, Nagoya 466-8555, Japan
seki@nitech.ac.jp

Abstract. We study the problem of mining frequent closed patterns in multi-relational databases in a distributed environment. In multi-relational data mining (MRDM), relational patterns involve multiple relations from a relational database, and they are typically represented in datalog language (a class of first order logic). Our approach is based on the notion of *iceberg query lattices*, a formulation of MRDM in terms of formal concept analysis (FCA), and we apply it to a distributed mining setting. We assume that a database considered contains a special predicate called *key*, which determines the entities of interest and what is to be counted, and that each datalog query contains an atom key, where variables in a query are linked to a given target object corresponding to the key. We show that the iceberg query lattice in this case can be defined similarly in the literature. Next, given two local databases (*horizontal* partitions) and their sets of closed patterns (concepts), we show that the subposition operator, which constructs a global Galois (concept) lattice from the direct product of two lattices studied in the literature, can be utilized to generate the set of closed patterns in the global database. The correctness of our algorithm is shown, and some preliminary experimental results using a MapReduce framework are also given.

1 Introduction

Multi-relational data mining (MRDM) has been extensively studied for more than a decade (e.g., [5, 6] and references therein). The research topics discussed in the conventional data mining have been considered in this more expressive framework of MRDM, where data and patterns (or queries) are represented in the form of logical formulae such as Datalog (a class of first order logic). In contrast to the traditional data mining dealing with rather simple patterns such as itemsets, the expressive formalism of MRDM allows us to use more complex and structured data in a uniform way, including trees and graphs in particular, and multi-relational patterns in general.

* This work was partially supported by JSPS Grant-in-Aid for Scientific Research (C) 24500171 and the Kayamori Foundation of Information Science Advancement.

ISBN 978–84–695–5252–0, Universidad de Málaga (Dept. Matemática Aplicada)

On the other hand, Formal Concept Analysis (FCA) has been developed as a field of applied mathematics based on a clear mathematization of the notions of concept and conceptual hierarchy [7]. It has attracted much interest from various application areas including, among others, data mining, knowledge acquisition and software engineering (e.g., [8]).

Stumme [20] has proposed the notion of *iceberg query lattices*, which combines the notions of the above two fields, i.e., MRDM and FCA; Iceberg query lattices combine the notion of frequent datalog queries in MRDM with iceberg concept lattices (or *frequent closed itemsets*) in FCA. Then, it has been shown that we can apply the "full arsenal" of FCA-based methods to frequent queries, thereby allowing us to mine and visualize relational association rules. Condensed representations such as closed patterns and free patterns in MRDM have been also studied in *c-armr* [4], and in RelLCM2 [9].

In this paper, we study the problem of mining closed queries in multi-relational data based on these precursors. We apply the notion of iceberg query lattices to a distributed mining setting. The assumption that a given dataset is distributed and stored in different sites is reasonable, because we will not be able to move local datasets into a centralized site due to too much data size and/or privacy concerns. We also assume that a database considered contains a special predicate called *key* (e.g., [3, 4]), and that each datalog query is supposed to contain an atom key, where variables in a query are *linked* to a given target object corresponding to the key. Using an key atom, we can define the notion of the frequency of a datalog query, since the key atom determines the entities of interest and what is to be counted. We show that the iceberg query lattice in this case can be defined similarly in the literature. Next, given two local databases (*horizontal* partitions) and their sets of closed queries (concepts), we show that the we can construct the set of closed queries in the global database, by using *subposition* operator [7, 23], which constructs a global Galois (concept) lattice from the direct product of two lattices. We also present some preliminary experimental results using a distributed framework of MapReduce [2].

The organization of the rest of this paper is as follows. After summarizing some basic notations and definitions in FCA in Sect. 2, we reconsider the notion of iceberg query lattices with key in Sect. 3. We then explain our approach to distributed closed query mining in MRDB in Sect. 4. In Section 5, we show the effectiveness of our method by some preliminary experimental results. Finally, we give a summary of this work in Section 6.

2 Preliminaries: Formal Concept Analysis

We assume that the reader is familiar with the basic notions of Formal Concept Analysis (FCA), which are found in [7]. However, we recall some of the important definitions and notations.

Definition 1. A *(formal) context* $\mathcal{K} = (O, A, I)$ consists of a set O of objects, a set A of attributes, and a binary relation $I \subseteq O \times A$.

The mapping $f : \mathcal{P}(O) \to \mathcal{P}(A)$ is given by $f(X) = \{a \in A \mid \forall o \in X : (o, a) \in I\}$. The mapping $g : \mathcal{P}(A) \to \mathcal{P}(O)$ is given by $g(Y) = \{o \in O \mid \forall a \in Y : (o, a) \in I\}$.

If it is clear from the context whether f or g is meant, then we abbreviate both $f(\cdot)$ and $g(\cdot)$ just by $'$. In particular, Y'' stands for $f(g(Y))$.

A *(formal) concept* is a pair (X, Y) with $X \subseteq O, Y \subseteq A,\ X' = Y$, and $Y' = X$. X is called *extent*, and Y is called *intent* of the concept. The set $\mathcal{C}_\mathcal{K}$ of all concepts of $\mathcal{K}$ together with the partial order $(X_1, Y_1) \leq (X_2, Y_2) \leftrightarrow X_1 \subseteq X_2$ (which is equivalent to $Y_1 \supseteq Y_2$) is called the *concept lattice* of $\mathcal{K}$. □

In FCA [7], a set of context-oriented operators has been studied, including *apposition/subposition* operators, and they are extensively studied by Valtchev and Missaoui [23, 24]. The following definitions and lemma are due to [23].

Definition 2. Let $\mathcal{K}_1 = (O_1, A, I_1)$ and $\mathcal{K}_2 = (O_2, A, I_2)$ be two contexts with the same set of attributes A. Then the context $\mathcal{K} = (O_1 \dot\cup O_2, A, I_1 \dot\cup I_2)$ is called the *subposition* of $\mathcal{K}_1$ and $\mathcal{K}_2$, denoted by $\mathcal{K} = \frac{\mathcal{K}_1}{\mathcal{K}_2}$.

Usually, the extent of $\mathcal{K}$ is set to the disjoint union (denoted by $\dot\cup$) of the involved context extents, and this constraint is suitable for our current study.

Let $\mathcal{K}_i$ $(i = 1, 2)$ be a context, and $\mathcal{L}_i$ the corresponding lattice. The direct product of a pair of lattices $\mathcal{L}_1$ and $\mathcal{L}_2$, denoted by $\mathcal{L}_\times = \mathcal{L}_1 \times \mathcal{L}_2$, is itself a lattice $\mathcal{L}_\times = \langle \mathcal{C}_{\mathcal{K}_\times}, \leq_\times \rangle$, where $\mathcal{C}_{\mathcal{K}_\times} = \mathcal{C}_{\mathcal{K}_1} \times \mathcal{C}_{\mathcal{K}_2}$, and $(c_1, c_2) \leq_\times (\bar c_1, \bar c_2) \Leftrightarrow c_1 \leq_{\mathcal{L}_1} \bar c_1$ and $c_2 \leq_{\mathcal{L}_2} \bar c_2$.

Any concept of $\mathcal{L}$ can be projected upon the concept lattice, $\mathcal{L}_1$ $(\mathcal{L}_2)$ by restricting its extent to the set of "visible" objects, e.g., those in O_1 (O_2), respectively. The resulting mapping constitutes an order homomorphism between $\mathcal{L}$ and the direct product [7].

Definition 3. The function $\varphi : \mathcal{C}_\mathcal{K} \to \mathcal{C}_{\mathcal{K}_\times}$ maps a concept from the global lattice into a pair of concepts of the partial lattices by splitting its extent over the partial context object sets O_1 and O_2:

$$\varphi((X, Y)) = ((X \cap O_1, (X \cap O_1)'),\ (X \cap O_2, (X \cap O_2)')).$$

From the above definitions, we have the following property [23]:

Lemma 1. [23] For any global concept $c = (X, Y)$ and its image $\varphi(c) = ((X_1, Y_1), (X_2, Y_2))$, it holds that $X = X_1 \cup X_2$ and $Y = Y_1 \cap Y_2$. □

Example 1. Consider a context $\mathcal{K}$ in Fig. 1 (upper left). The concept lattice $\mathcal{C}_\mathcal{K}$ derived from $\mathcal{K}$ is shown in the right. Let $\mathcal{K}_1, \mathcal{K}_2$ (lower right of the figure) be a horizontal decomposition of $\mathcal{K}$, where $O_1 = \{1, 2\}$ and $O_2 = \{3, 4\}$. Then, $\mathcal{K}$ is the subposition of $\mathcal{K}_1$ and $\mathcal{K}_2$, i.e., $\mathcal{K} = \frac{\mathcal{K}_1}{\mathcal{K}_2}$. The concept lattices $\mathcal{C}_{\mathcal{K}_1}$ $(\mathcal{C}_{\mathcal{K}_2})$ derived from $\mathcal{K}_1$ $(\mathcal{K}_2)$ are shown in the right, respectively.

Consider a global concept $c = (123, d)$ in $\mathcal{C}_\mathcal{K}$. Then, $\varphi(c) = ((12, bd), (3, acd))$, and we have from Lemma 1 that $\{123\} = \{12\} \cup \{3\}$ and $\{d\} = \{bd\} \cap \{acd\}$. □

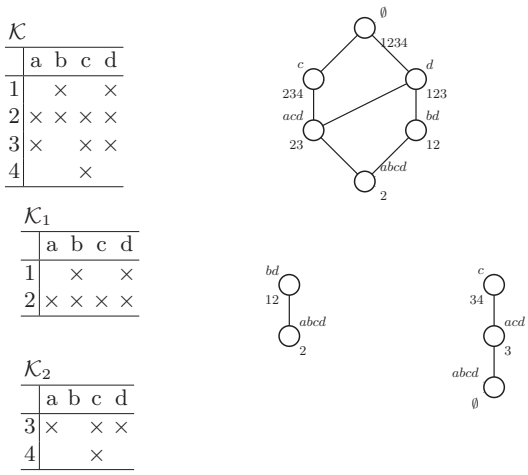

Fig. 1. Upper: A Context $\mathcal{K}$ and the Hasse diagram of the concept lattice derived from $\mathcal{K}$. Lower: A horizontal decomposition $\mathcal{K}_1, \mathcal{K}_2$ of $\mathcal{K}$, and the Hasse diagrams of the concept lattices derived from $\mathcal{K}_1, \mathcal{K}_2$.

FCA provides a framework for frequent itemset mining (FIM), where the intent of a concept corresponds to a closed itemset. The subposition operator will be readily used for mining frequent closed itemsets (FCIs) in a global transaction database $\mathcal{D}$ from the local FCIs from two disjoint (horizontal) partitions $\mathcal{D}_1$ and $\mathcal{D}_2$, provided that we mine all the partitions with an (absolute) support being set to 1, i.e. when we consider as frequent any itemset which occur at least once in $\mathcal{D}$. In fact, Lucchese et al. [15] show the following property:

Theorem 1 (Lucchese et al. [15]). Let $\mathcal{D}$ be transaction database, and $\mathcal{D}_1$, $\mathcal{D}_2$ two disjoint (horizontal) partitions of $\mathcal{D}$. Let $\mathcal{C}$ be the set of FCIs of $\mathcal{D}$, and $\mathcal{C}_1$ ($\mathcal{C}_2$) the set of local FCIs of $\mathcal{D}_1$ ($\mathcal{D}_2$), respectively. Then, $\mathcal{C}$ is computed from $\mathcal{C}_1$ and $\mathcal{C}_2$ as $\mathcal{C} = (\mathcal{C}_1 \cup \mathcal{C}_2) \cup \{C_1 \cap C_2 \mid (C_1, C_2) \in (\mathcal{C}_1 \times \mathcal{C}_2)\}$. □

Namely, $\mathcal{C}$ is obtained by collecting the closed itemsets contained in $\mathcal{C}_1$ and $\mathcal{C}_2$, and intersecting them to obtain further ones. It is easy to see that this exactly corresponds to Lemma 1 based on the subposition operator. In the following, we will apply the subposition operator to a more expressive framework of MRDM.

3 Iceberg Query Lattices in Multi-Relational DM

3.1 Multi-Relational Data Mining

In the task of frequent pattern mining in multi-relational databases, we assume that we have a given database **r**, a language of patterns, and a notion of frequency which measures how often a pattern occurs in the database. We use Datalog

Customer
key
allen
carol
diana
fred

Parent	
SR.	*JR.*
allen	bill
allen	jim
carol	bill
diana	eve
fred	eve
fred	hera

Buys	
key	*item*
allen	pizza
carol	pizza
diana	cake
fred	cake

Male
person
bill
jim

Female
person
eve
hera

Fig. 2. An Example of Datalog Database **r** with customer relation as a key

to represent data and patterns. We assume some familiarity with the notions of logic programming (e.g., [14, 16]), although we introduce some notions and terminology in the following.

An *atom* (or *literal*) is an expression of the form $p(t_1, \ldots . t_n)$, where p is a *predicate* (or *relation*) of arity n, denoted by p/n, and each t_i is a *term*, i.e., a constant or a variable.

A substitution $\theta = \{X_1/t_1, \ldots, X_n/t_n\}$ is an assignment of terms to variables. The result of applying a substitution θ to an expression E is the expression $E\theta$, where all occurrences of variables V_i have been simultaneously replaced by the corresponding terms t_i in θ. The set of variables occurring in E is denoted by $Var(E)$.

A *pattern* is expressed as a conjunction of atoms (literals) $l_1 \wedge \cdots \wedge l_n$, denoted simply by $l_1, \ldots, l_n$. A pattern is sometimes called a *query*. Let C be a pattern (i.e., a conjunction) and θ a substitution of $Var(C)$. When $C\theta$ is logically entailed by a database **r**, we write it by $\mathbf{r} \models C\theta$. Let $answerset(C, \mathbf{r})$ be the set of substitutions satisfying $\mathbf{r} \models C\theta$. We will represent conjunctions in list notation, i.e., $[l_1, \ldots, l_n]$. For a conjunction C and an atom p, we denote by $[C, p]$ the conjunction that results from adding p after the last element of C.

In multi-relational data mining, one of predicates is often specified as a *key* (or *target*), which determines the entities of interest and what is to be counted.

Example 2. Let **r** be a multi-relational DB in Fig. 2, which consists of five relations, including Customer, Parent, Buys and so on. For each relation, we introduce a corresponding predicate, e.g., *customer* for relation Customer.

Let P be a pattern of the form: $customer(X), parent(X, Y), buys(X, pizza)$. $P\theta$ is logically entailed by **r**, if there exists a tuple (a_1, a_2) such that $a_1 \in$ Customer, $(a_1, a_2) \in$ Parent, and $(a_1, pizza) \in$ Buys. Then, $answerset(P, \mathbf{r}) = \{\{X/allen, Y/bill\}, \{X/allen, Y/jim\}, \{X/carol, Y/bill\}\}$. □

As explained in Sect. 1, in a typical task of MRDM, a user is usually expected to specify a special predicate *key* (or *target*) (e.g., [3, 4]). The key is an atom which determines the entities of interest and what is to be counted. The key (target) is thus to be present in all patterns considered. In Example 2, the key is predicate *customer*.

A pattern containing a key is not always meaningful to be mined. For example, let $C = [customer(X), parent(X, Y), buys(Z, pizza)]$ be a conjunction in Example 2. Variable Z in C is not *linked* to variable X in key atom $customer(X)$; an object represented by Z will have nothing to do with key object X. It will be inappropriate to consider such a conjunction as an intended pattern to mine. In ILP, the following notion of *linked literals* [10] is a standard one to specify the so-called *language bias*.

Definition 4 (Linked Literal). [10] Let $key(X)$ be a key atom and l a literal. l is said to be *linked* to $key(X)$, if either $X \in Var(l)$ or there exists a literal l_1 such that l is linked to $key(X)$ and $Var(l_1) \cap Var(l) \neq \emptyset$. □

Given a database $\mathbf{r}$ and a key atom $key(X)$, we assume that there are predefined finite sets of predicate (resp. variables; resp. constant symbols), and that, for each literal l in a conjunction C, it is constructed using the predefined sets. Moreover, each pattern C of conjunctions to be mined satisfies the following conditions: $key(X) \in C$ and, for each $l \in C, l$ is linked to $key(X)$. In the following, we denote by $\mathcal{Q}$ the set of queries (or patterns) satisfying the above bias condition.

Let $\mathbf{r}$ be a database and Q be a query containing a key atom $key(X)$. Then, the *support* (or *frequency*) of C, denoted by $supp(Q, \mathbf{r}, key)$, is defined as:

$$supp(Q, \mathbf{r}, key) = \frac{|\{\theta_{key} \mid \theta \in answerset(Q, \mathbf{r})\}|}{|answerset(key(X), \mathbf{r})|},$$

where θ_{key} is the *restriction* of $\theta = \{X/t, \dots\}$ w. r. t. $key(X)$, defined by $\theta_{key} = \{X/t\}$ for some term t. The numerator in the above formula is called the *support count* (or *absolute support*). Q is said to be *frequent*, if $supp(Q, \mathbf{r}, key)$ is no less than some user defined threshold min_sup.

3.2 Iceberg Query Lattices with Key

We now consider the notion of a formal context in MRDM, following [20].

Definition 5. [20] Let $\mathbf{r}$ be a datalog database and $\mathcal{Q}$ a set of datalog queries. The *formal context associated to* $\mathbf{r}$ and $\mathcal{Q}$ is defined by $\mathcal{K}_{\mathbf{r},\mathcal{Q}} = (O_{\mathbf{r},\mathcal{Q}}, A_{\mathbf{r},\mathcal{Q}}, I_{\mathbf{r},\mathcal{Q}})$, where $O_{\mathbf{r},\mathcal{Q}} = \{\theta \mid \theta$ is a grounding substitution for all $Q \in \mathcal{Q}\}$, and $A_{\mathbf{r},\mathcal{Q}} = \mathcal{Q}$, and $(\theta, Q) \in I_{\mathbf{r},\mathcal{Q}}$ if and only if $\theta \in answerset(Q, \mathbf{r})$. □

Each $\theta \in answerset(Q, \mathbf{r})$ is often called an *occurrence* of Q in $\mathbf{r}$. We denote by $\mathcal{O}(Q; \mathbf{r})$ the set of the occurrences of Q in $\mathbf{r}$, namely, $\mathcal{O}(Q; \mathbf{r}) = answerset(Q, \mathbf{r})$.

From this formal context, we can define the concept lattice the same way as in [20]. We first introduce an equivalence relation $\sim_{\mathbf{r}}$ on the set of queries: Two queries Q_1 and Q_2 are said to be *equivalent* with respect to database $\mathbf{r}$ if and only if $answerset(Q_1, \mathbf{r}) = answerset(Q_2, \mathbf{r})$.

Definition 6 (Closed Query). Let $\mathbf{r}$ be a datalog database and $\sim_{\mathbf{r}}$ the equivalence relation on a set of datalog queries $\mathcal{Q}$. A query (or pattern) Q is said to be *closed* (w.r. t. $\mathbf{r}$ and $\mathcal{Q}$), iff Q is the most specific query among the equivalence class to which it belongs: $\{Q_1 \in \mathcal{Q} \mid Q \sim_{\mathbf{r}} Q_1\}$. □

For any query Q_1, its *closure* is a closed query Q such that Q is the most specific query among $\{Q \in \mathcal{Q} \mid Q \sim_{\mathbf{r}} Q_1\}$. Since it uniquely exists, we denote it by $Clo(Q_1; \mathbf{r})$. Note that $Var(Q_1) = Var(Clo(Q_1; \mathbf{r}))$ by definition. We refer to this as the *range-restricted* condition here.

Stumme [20] showed that the set of frequent closed queries forms a lattice. In our framework, it is necessary to take our bias condition into consideration. To do that, we employ the well-known notion of the most specific generalization (or *least generalization*) [18, 16].

For queries Q_1 and Q_2, we denote by $lg(Q_1, Q_2)$ the least generalization of Q_1 and Q_2. Moreover, the *join* of Q_1 and Q_2, denoted by $Q_1 \vee Q_2$, is defined as: $Q_1 \vee Q_2 = lg(Q_1, Q_2)|_{\mathcal{Q}}$, where, for a query Q, $Q|_{\mathcal{Q}}$ is the *restriction* of Q to $\mathcal{Q}$, defined by a conjunction consisting of every literal l in Q which is linked to $key(X)$, i.e., deleting every literal in Q not linked to $key(X)$.

Definition 7. [20] Let $\mathbf{r}$ be a datalog database and $\mathcal{Q}$ a set of datalog queries. The *iceberg query lattice associated to* $\mathbf{r}$ and $\mathcal{Q}$ for *minsupp* $\in [0, 1]$ is defined as: $\mathcal{C}_{\mathbf{r},\,\mathcal{Q}} = (\{Q \in \mathcal{Q} \mid Q \text{ is closed w.r.t. } \mathbf{r} \text{ and } \mathcal{Q}, \text{ and } Q \text{ is frequent}\}, \models)$, where $\models$ is the usual logical implication. □

Theorem 2. Let $\mathbf{r}$ be a datalog database and $\mathcal{Q}$ a set of datalog queries where all queries contain an atom *key* and they are linked. Then, $\mathcal{C}_{\mathbf{r},\,\mathcal{Q}}$ is a $\vee$-semi-lattice.

Proof. (Sketch) Let Q_1, Q_2 be frequent closed queries in $\mathcal{Q}$. Then, it is easy to see that their least generalization $lg(Q_1, Q_2)$ is closed and frequent. However, it might not be linked to $key(X)$. For example, consider that Q_1 (Q_2) is of the form: $Q_1 = key(X), p(X, Y), m(Y)$ ($Q_2 = key(X), q(X, Y), m(Y)$), respectively. Then, $lg(Q_1, Q_2) = key(X), m(Y)$, which is not linked to $key(X)$, although it is a closed query. In this case, $Q_1 \vee Q_2 = lg(Q_1, Q_2)|_{\mathcal{Q}} = key(X)$, which satisfies the bias condition from the definition. We can show that the resulting $Q_1 \vee Q_2$ is in fact a closed query in the sense of Def. 6. □

Example 3. Continued from Example 2. Fig. 3 shows the iceberg query lattice associated to $\mathbf{r}$ in Ex. 2 and $\mathcal{Q}$ with the support count 1, where each query $Q \in \mathcal{Q}$ has $customer(X)$ as a key atom, denoted by $key(X)$ for short, $Var(Q) \subseteq \{X, Y\}$ and the 2nd argument of predicate *buys* is a constant. □

4 Distributed Closed Pattern Mining in MRDB

Our purpose in this work is to mine global concepts in a distributed setting, where a global database is supposed to be horizontally partitioned appropriately, and stored possibly in different sites. Our approach is to first perform the

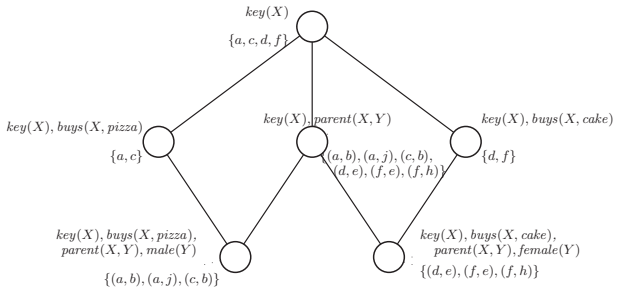

Fig. 3. The Iceberg Query Lattice Associated to $\mathbf{r}$ in Ex. 2: In the figure, a substitution $\theta = \{X/t_1, Y/t_2\}$ (resp., $\theta = \{X/t_1\}$) in an occurrence set is denoted simply by (t_1, t_2) (resp., t_1). The name of each person in $\mathbf{r}$ is abbreviated to its first character.

computations of local concepts on each partition of the global DB, and then combine the local concepts by using the subposition operator.

4.1 Horizontal Decomposition of MRDB

We first consider the notion of a *horizontal decomposition* of a multi-relational DB. Since a multi-relational DB consists of multiple relations, its horizontal decomposition is not immediately clear.

Definition 8. Let $\mathbf{r}$ be a multi-relational datalog database with a key predicate *key*. We call a pair $\mathbf{r}_1, \mathbf{r}_2$ a *horizontal decomposition* of $\mathbf{r}$, if

1. $\mathrm{key}_{\mathbf{r}} = \mathrm{key}_{\mathbf{r}_1} \cup \mathrm{key}_{\mathbf{r}_2}$, i.e., the key relation $\mathrm{key}_{\mathbf{r}}$ in $\mathbf{r}$ is disjointly decomposed into $\mathrm{key}_{\mathbf{r}_1}$ and $\mathrm{key}_{\mathbf{r}_2}$ in $\mathbf{r}_1$ and $\mathbf{r}_2$, respectively, and
2. for any query Q, $\mathit{answerset}(Q, \mathbf{r}) = \mathit{answerset}(Q, \mathbf{r}_1) \cup \mathit{answerset}(Q, \mathbf{r}_2)$. □

The second condition in the above states that the relations other than $\mathrm{key}_{\mathbf{r}}$ are decomposed so that any answer substitution in $\mathit{answerset}(Q, \mathbf{r})$ is computed either in $\mathbf{r}_1$ or $\mathbf{r}_2$, thereby being preserved in this horizontal decomposition.

Given a horizontal decomposition of a multi-relational DB, we can utilize any preferable concept (or closed pattern) mining algorithm for computing local concepts on each partition, as long as the mining algorithm is applicable to MRDM and its resulting patterns satisfy our bias condition. For example, Stumme [20] discussed the algorithm called TITANIC [21], which is based on a level-wise approach. We use here an algorithm called ffLCM [19], which is based on the notion of *closure extension* due to Pasquier et al. [17] in FIM, and then elaborated by Uno et al. [22].

4.2 Subposition Operator in MRDM

We now present the counterpart to Lemma 1 in closed pattern mining in MRDB. We first modify the mapping φ in Def. 3 suitably for our purpose.

Definition 9. Let $\mathbf{r}$ be a datalog database, and $\mathbf{r}_1, \mathbf{r}_2$ a horizontal decomposition of $\mathbf{r}$. Let $(\mathcal{O}(Q;\mathbf{r}), Q)$ be a concept in $\mathbf{r}$, i.e., Q is a closed query and $\mathcal{O}(Q;\mathbf{r}) = \mathit{answerset}(Q,\mathbf{r})$. Then,

$$\tilde{\varphi}((\mathcal{O}(Q;\mathbf{r}), Q)) = ((\mathcal{O}(Q;\mathbf{r}_1),\ \mathit{Clo}(Q;\mathbf{r}_1)),\ (\mathcal{O}(Q;\mathbf{r}_2),\ \mathit{Clo}(Q,\mathbf{r}_2))).$$

To give the counterpart to Lemma 1 in MRDM, we need another definition of join. Let Q_1 and Q_2 be queries which contain the same set $\mathcal{V}$ of variables, i.e., $\mathit{Var}(Q_1) = \mathit{Var}(Q_2) = \mathcal{V}$. We define $Q_1 \vee_{RR} Q_2 = \mathit{lg}(Q_1, Q_2)|_{\mathcal{V},\mathcal{Q}}$, where, for a query Q, $Q|_{\mathcal{V},\mathcal{Q}}$ is the *restriction* of Q to $\mathcal{V}$ and $\mathcal{Q}$, defined by a conjunction consisting of every literal l in Q such that $\mathit{Var}(l) \subseteq \mathcal{V}$ and l is linked to *key*, i.e., $Q|_{\mathcal{V},\mathcal{Q}}$ is constructed from Q by deleting every literal in Q which contains a variable not in $\mathcal{V}$, then deleting every remaining literal not linked to *key*.

Theorem 3. Let $\mathbf{r}$ be a datalog database, and $\mathbf{r}_1, \mathbf{r}_2$ a horizontal decomposition of $\mathbf{r}$. For any global concept $c = (\mathcal{O}(Q;\mathbf{r}), Q)$ in $\mathbf{r}$, and its image $\tilde{\varphi}(c) = ((\mathcal{O}(Q_1;\mathbf{r}_1), Q_1), (\mathcal{O}(Q_2;\mathbf{r}_2), Q_2))$, it holds that

$$\mathcal{O}(Q;\mathbf{r}) = \mathcal{O}(Q_1;\mathbf{r}_1) \cup \mathcal{O}(Q_2;\mathbf{r}_2) \ \text{ and } \ Q = Q_1 \vee_{RR} Q_2.$$

Remark 1. We omit the proof here, since we can prove the theorem similarly to [23]. Instead, we give an example which will be helpful to understand why we need an extra provision for considering the least generalization in this case.

Let Q be a query of the form: $\mathit{key}(A), p(A,B)$. Suppose that Q_1 (Q_2) is a query of the form: $Q_1 = \mathit{key}(A), p(A,B), q(A,B)$ ($Q_2 = \mathit{key}(A), p(A,B), q(B,A)$), respectively. Then, $\mathit{lg}(Q_1, Q_2) = \mathit{key}(A), p(A,B), q(C,D)$, where C and D are newly introduced variables in the least generalization. In this case, since $\mathit{Var}(Q) = \mathit{Var}(Q_1) = \mathit{Var}(Q_2) = \{A,B\}$, $Q_1 \vee_{RR} Q_2$ is $\mathit{key}(A), p(A,B)$, which coincides with Q.

Finally, we note that, in the case of transaction databases, the above theorem coincides with Theorem 1 in Sect. 2. □

Example 4. Continued from Example 3. We consider a horizontal decomposition $\mathbf{r}_1, \mathbf{r}_2$ of $\mathbf{r}$ such that the key relation $\mathrm{key}_{\mathbf{r}}$ (i.e., Customer) in $\mathbf{r}$ is decomposed into $\mathrm{key}_{\mathbf{r}_1} = \{\mathrm{allen}, \mathrm{carol}\}$ and $\mathrm{key}_{\mathbf{r}_2} = \{\mathrm{dian}, \mathrm{fred}\}$, and the other relations than Customer are decomposed so that they satisfy the second condition of Def. 8.

Let Q be a pattern of the form: $[\mathit{key}(X), \mathit{parent}(X,Y)]$ in Fig. 3. We have that $Q_1 = \mathit{Clo}(Q;\mathbf{r}_1) = [Q, \mathit{buys}(X, \mathit{pizza}), \mathit{male}(Y)]$, and $Q_2 = \mathit{Clo}(Q;\mathbf{r}_2) = [Q, \mathit{buys}(X, \mathit{cake}), \mathit{female}(Y)]$. Then, it holds that $Q = Q_1 \vee_{RR} Q_2$. □

5 Distributed Mining Using MapReduce Framework

Since the computation of local concepts can be done independently, it is expected that our algorithm is amenable to data-parallelism. We have therefore implemented our algorithm using MapReduce framework [2], although any framework supporting data-parallelism will do for our purpose.

In MapReduce framework, the user expresses the computation in terms of two functions: *map* and *reduce*. The map function takes an input key/value pair and produces a set of intermediate key/value pairs. Then, the set of intermediate key/value pairs are passed to the reduce function. The reduce function accepts an intermediate key and a set of values for that key, and it then merges (or aggregate) these values together to form a possibly smaller set of values.

However, our use of MapReduce framework is very simple; We use map operation to each local DB to compute a set of its local concepts. An intermediate key/value pair simply consists of $(DB\_id, \mathcal{C}_{id})$, where $\mathcal{C}_{id}$ is the set of local concepts of DB_id. We then apply a reduce operation which simply combines the derived results to form an input to the subsequent subposition operator. We thus simply exploited map for computing local concepts independently. We employed Hadoop[1],an open source implementation of MapReduce.

We now present some preliminary results of our experiments. We implemented our algorithm by using Java 1.6.0_22. Experiments were performed on 6 PCs with Intel Core i5 processors running at 2.8GHz, 8GB of main memory, and 8MB of L2 cache, working under Ubuntu 11.04. We used Hadoop 0.20.2 using 6 PCs, and 2 mappers working on each PC.

Fig. 4 summarizes the results of the execution time for a test data on the mutagenicity prediction,[2] containing 30 chemical compounds. Each compound is represented by a set of facts using predicates such as *atom*, *bond*, for example. The size of the set of predicate symbols is 12. The size of key atom ($active(X)$) is 230, and minimum support $min\_sup = \frac{1}{230}$. We assume that patterns contain at most 4 variables and they contain no constant symbols. The number of the concepts mined is $4,831$.

Fig. 4 shows that the execution times t_1 for mining local concepts are reduced almost linearly with the number of partitions from 1 (i.e., no partitioning) to 8. When the number of partitions is 16, the speed-up did not scale well compared to the other cases. This is a reasonable result; Due to the restriction of our current experiment environment, we used 6 PCs. Therefore, at most 12 mappers are simultaneously available. On the other hand, the execution times t_2 for merging local concepts to obtain global concepts increase almost linearly with the number p of partitions from 1 (i.e., no partitioning) to 16. This is also reasonable; the number of subposition operators applied is $(p-1)$ when we have p partitions.

[1] Hadoop: Open source implementation of MapReduce. http://lucene.apache.org/hadoop/.

[2] http://www.comlab.ox.ac.uk/activities/machinelearning/mutagenesis.html

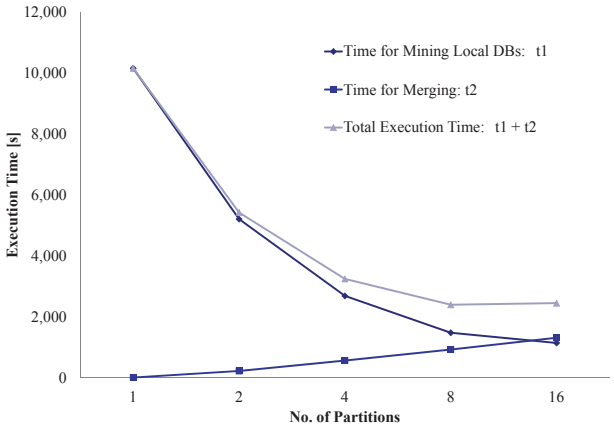

Fig. 4. Execution Time

6 Concluding Remarks

We have studied the problem of mining frequent closed patterns in multi-relational databases in a distributed environment. To do that, we have first reconsidered the notion of iceberg query lattices, where each datalog query contains an atom key, and the variables in a query are linked to the key. We have then proposed the notion of a horizontal decomposition of a given MRDB, and explained how the subposition operator can be utilized to generate the set of closed queries in the global database from the two sets of local closed queries in the two partitions. We have exemplified the effectiveness of our method by some preliminary experimental results using Hadoop.

As discussed in [1], efficiency and scalability have been major concerns in MRDM. Krajca et al. [11, 12] have proposed algorithms which allow us to compute search trees for concepts simultaneously either in parallel or in a distributed manner. Since their approaches are orthogonal to ours, it would be beneficial to employ their algorithms for computing local concepts in our method.

In this work, we have confined ourselves to horizontal partitions of a global context. It will be interesting to study *vertical* partitioning and their mixture in MRDM, where the apposition operator studied by Valtchev et al. [24] will play an important role. Our future work includes developing an efficient algorithm for handling such a general case, as well as accumulating more experimental results on different MRDBs to confirm the effectiveness of our subposition operator.

Acknowledgement The authors would like to thank anonymous reviewers for their useful comments on our paper. The authors are grateful to Seiji Yamazaki for preparing the experiments in this paper.

References

1. Blockeel, H., Sebag, M.: Scalability and efficiency in multi-relational data mining. SIGKDD Explorations Newsletter 2003, Vol.4, Issue 2, pp.1-14 (2003)
2. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. Commun. ACM, Vol. 51, No. 1, pp.107–113, 2008.
3. Dehaspe, L.: Frequent pattern discovery in first-order logic, PhD thesis, Dept. Computer Science, Katholieke Universiteit Leuven, 1998.
4. De Raedt, L., Ramon, J.: Condensed representations for Inductive Logic Programming. In: Proc. KR'04, pp. 438-446 (2004)
5. Dzeroski, S.: Multi-Relational Data Mining: An Introduction. SIGKDD Explorations Newsletter 2003, Vol.5, Issue 1, pp.1-16 (2003)
6. Dzeroski, S., Lavrač, N. (eds.): Relational Data Mining. Springer-Verlag, Inc. 2001.
7. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, 1999.
8. Ganter, B., Stumme, G., Wille, R.: Formal Concept Analysis, Foundations and Applications. LNCS 3626, Springer, 2005.
9. Garriga,G. C., Khardon, R., De Raedt, L.: On Mining Closed Sets in Multi-Relational Data. IJCAI 2007, pp.804-809 (2007)
10. Helft, N.: Induction as nonmonotonic inference. In Proc. KR'89, pp. 149–156, 1989.
11. Krajca, P., Vychodil, V.: Distributed Algorithm for Computing Formal Concepts Using Map-Reduce Framework, IDA '09, Springer-Verlag, pp. 333–344, 2009.
12. Krajca, P., Outrata, J., Vychodil, V.: Parallel algorithm for computing fixpoints of Galois connections, AMAI, Vol. 59, No. 2, pp. 257–272, Kluwer Academic Pub., 2010.
13. Kuznetsov, S. O., Obiedkov, S. A.: Comparing performance of algorithms for generating concept lattices. J. Exp. Theor. Artif. Intell., 14(2-3):189.216, 2002.
14. Lloyd, J. W.: Foundations of Logic Programming, Springer, 1987, Second edition.
15. Lucchese, C., Orlando, S., Rergo, R.: Distributed Mining of Frequent Closed Itemsets: Some Preliminary Results. International Workshop on High Performance and Distributed Mining (2005).
16. Nienhuys-Cheng, S-H., de Wolf, R.: Foundations of Inductive Logic Programming, LNAI 1228, Springer, 1997.
17. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Discovering Frequent Closed Itemsets for Association Rules. Proc. ICDT'99, LNAI 3245, pp. 398-416 (1999)
18. Plotkin, G.D.: A Note on Inductive Generalization. Machine Intelligence, Vol. 5, pp. 153-163, 1970.
19. Seki, H., Honda, Y., Nagano, S.: On Enumerating Frequent Closed Patterns with Key in Muti-relational Data. LNAI 6332, pp. 72-86 (2010)
20. Stumme, G.: Iceberg Query Lattices for Datalog. In Conceptual Structures at Work, LNCS 3127, Springer-Verlag, pp. 109-125, 2004.
21. Stumme, G., Taouil, R., Bastide, Y., Pasquier, N., Lakhal, L.: Computing Iceberg Concept Lattices with Titanic. J. KDE 42(2), 2002, pp. 189-222.
22. Uno, T., Asai, T. Uchida, Y., Arimura, H.: An Efficient Algorithm for Enumerating Closed Patterns in Transaction Databases. Proc. DS'04, LNAI 3245, pp. 16-31 (2004)
23. Valtchev, P., Missaoui, R.: Building Concept (Galois) Lattices from Parts: Generalizing the Incremental Methods. In Proc. of the 9th Int'l. Conf. on Conceptual Structures: Broadening the Base (ICCS '01), Springer-Verlag, London, UK, 290-303.
24. Valtchev, P., Missaoui, R., Pierre Lebrun, P.: A Partition-based Approach towards Constructing Galois (Concept) Lattices. Discrete Mathematics 256(3): 801-829 (2002)