ERATO セミナ 2010 - No. 12

分散データベースからの k 頻出飽和アイテム集合のプライバシー保護発見

山本章博
京都大学 大学院情報学研究科

2010/8/6

概要

水平分割され分散配置されたトランザクションデータベースに，全体として頻出する飽和アイテム集合を暗号化と組み合わせて発見する手続きを提案する．頻出飽和アイテム集合は，頻出アイテム集合の圧縮形とみなせることから，分散配置されたデータベースからのプライバシー保護発見に有用と考えられる．本講では基本アイデアを実験結果とともに報告する．

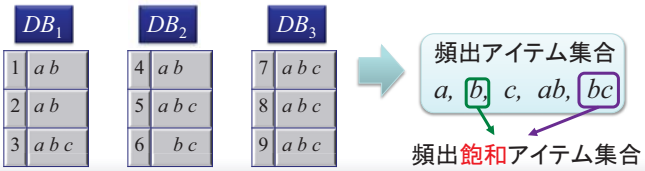
分散データベースからの頻出飽和アイテム集合のプライバシー保護発見

06 Aug. 2010　ERATO セミナ

京都大学情報学研究科
久野慎弥*　土井晃一郎**　山本章博
*現在　野村総合研究所
**現在 東京大学新領域創成科学研究科

概要

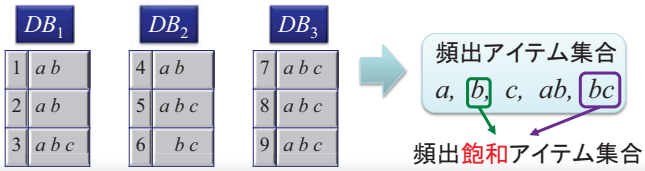
問題：**大規模データベースから興味深いパターンを発見すること**
- “データベース”・・・分散配置されたトランザクションDB
  - トランザクションDB：バスケット分析（買い物かごや取引レコードで アイテム購入の相関ルールを発見する問題）で用いられるモデル
- “興味深いパターン”・・・出現頻度の高いアイテム集合（頻出アイテム集合）の中で代表的なもの（頻出飽和アイテム集合）



概要

問題：**大規模データベースから興味深いパターンを発見すること**
- “データベース”・・・分散配置されたトランザクションDB
  - トランザクションDB：バスケット分析（買い物かごや取引レコードで アイテム購入の相関ルールを発見する問題）で用いられるモデル
- 頻出

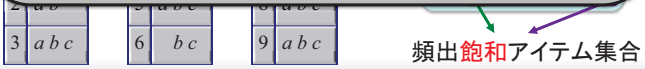
本研究では，分散配置されたトランザクションDBから頻出飽和アイテム集合を発見することを目標とする．



概要

問題：**大規模データベースから興味深いパターンを発見すること**

- 分散配置されたデータベースから，プライバシ保護のため暗号化を用いて頻出飽和アイテム集合をすべて発見する手続きSFC-Combineを構築
  - 通信を考慮しない手続きFC-Combineを構築し，これを基にする
- 従来研究の手続きとの性能比較を行い，従来より通信コストを抑えられていることを確認



発表の概要

- 本研究の背景
- 準備
  - 頻出アイテム集合
  - 頻出飽和アイテム集合
  - 問題例
- マイニングの手続き
  - FC-Combine
  - SFC-Combine
- 性能比較
- 今後の課題

研究の背景

- ネットワーク上に分散されたデータベースから興味深いパターンを発見する手法に近年注目が集まっているが，幾つかの問題点が指摘されている．
  - 分散データベースに蓄えられているデータ量の増大ペースに対して，ネットワーク帯域のコストパフォーマンス改善が追い付いていない．[SCS2009 Konishi]

⇒ 通信コストを削減して全体として蓄えられたデータを分析する発見する手法の必要性

  - データベースから個人情報が漏洩することの危険性が大きいが，個々のデータベースを死蔵させることの損失もまた大きい．[Kantarcioglu et al.][Verykios et al.]

⇒ プライバシを保護してデータマイニングを行う手法の必要性

頻出アイテム集合発見問題

- 頻出アイテム集合発見問題・・・データマイニング研究の発展の契機となった問題

DB

1	a c d
2	b c e
3	a b c e
4	b e
5	a b c e

minsup = 50%

頻出アイテム集合発見問題

与えられたトランザクションデータベースDBと最小支持度minsupに対して, 出現頻度がminsup以上であるアイテム集合を全て発見せよ.

アイテム集合Xの出現頻度 ＝ $\frac{X\text{が出現するトランザクションの数}}{DB\text{のトランザクションの数}}$

頻出アイテム集合発見問題

- 頻出アイテム集合発見問題・・・データマイニング研究の発展の契機となった問題

DB

1	a c d
2	b c e
3	a b c e
4	b e
5	a b c e

minsup = 50%

頻出アイテム集合の例:

a (sup: 60%) *b* (sup: 80%)
ac (sup: 60%) *bc* (sup: 60%)
be (sup: 80%) *ce* (sup: 60%)
bce (sup: 60%)
・・・

($\{i_1, i_2, \ldots, i_k\}$ を$i_1 i_2 \ldots i_k$と表す)

頻出アイテム集合発見問題

- 頻出アイテム集合発見問題・・・データマイニング研究の発展の契機となった問題

DB

1	a c d
2	b c e
3	a b c e
4	b e
5	a b c e

minsup = 50%

頻出アイテム集合の例:

アイテムの種類に対して指数オーダー数の解が存在する
→出力を絞り込むことが必要
[Uno et al. 05]

($\{i_1, i_2, \ldots, i_k\}$ を$i_1 i_2 \ldots i_k$と表す)

頻出飽和アイテム集合

アイテム集合Cが飽和アイテム集合である
⇕
Cを含むトランザクション全てに共通して含まれるアイテム集合がC自身である.

DB

1	a c d
2	b c e
3	a b c e
4	b e
5	a b c e

minsup = 50%

頻出飽和アイテム集合: *c, ac, be, bce*
頻出であるが飽和でない例: *a, bc, …*

頻出飽和アイテム集合

アイテム集合Cが飽和アイテム集合である
⇕
Cを含むトランザクション全てに共通して含まれるアイテム集合がC自身である.

DB

1	a c d
2	b c e
3	a b c e
4	b e
5	a b c e

minsup = 50%

頻出飽和アイテム集合: *c, ac, be, bce*
頻出であるが飽和でない例: *a, bc, …*

頻出飽和アイテム集合

アイテム集合Cが飽和アイテム集合である
⇕
Cを含むトランザクション全てに共通して含まれるアイテム集合がC自身である.

DB

1	a c d
2	b c e
3	a b c e
4	b e
5	a b c e

minsup = 50%

頻出飽和アイテム集合: *c, ac, be, bce*
頻出であるが飽和でない例: *a, bc, …*

頻出飽和アイテム集合

DB	
1	a c d
2	b c e
3	a b c e
4	b e
5	a b c e

minsup = 50%

アイテム集合Cが飽和アイテム集合である

⇕

Cを含むトランザクション全てに共通して含まれるアイテム集合がC自身である.

頻出飽和アイテム集合: c, ac, be, bce

頻出であるが飽和でない例: $a, bc, \ldots$

頻出飽和アイテム集合

DB	
1	a c d
2	b c e
3	a b c e
4	b e
5	a b c e

minsup = 50%

アイテム集合Cが飽和アイテム集合である

⇕

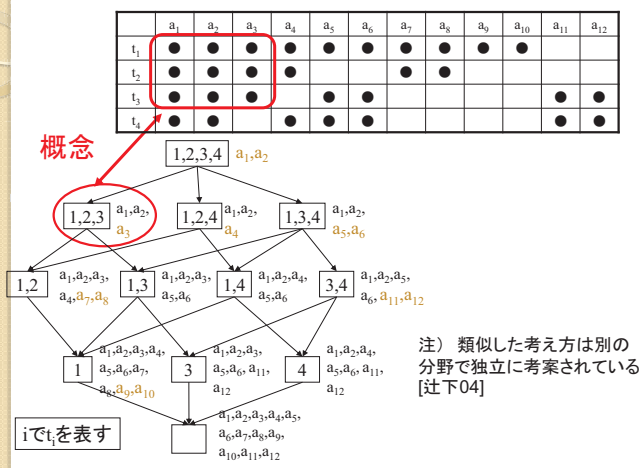
Cを含むトランザクション全てに共通して含まれるアイテム集合がC自身である.

頻出飽和アイテム集合: c, ac, be, bce

頻出であるが飽和でない例: $a, bc, \ldots$

補題: すべての頻出アイテム集合は頻出飽和アイテム集合から求めることができる.

Formal Concept Analysis [Wille 84]



注） 類似した考え方は別の分野で独立に考案されている [辻下04]

用いるモデル

- アイテム全体の集合をIとする.
- アイテムの集合とトランザクションIDの組をトランザクションとよび, トランザクションの集合をトランザクションDBとよぶ.

- 分散DBのモデルとして, $DB_i \cap DB_j = \varphi$ なるN個のトランザクションDB, $DB_1, DB_2, \ldots, DB_N$を用いる.
- $DB_1, DB_2, \ldots, DB_N$を部分DBとよび, DB_iを管理するサーバをサイトiとよぶ.
- $DB_{1\ldots N} = DB_1 \cup \ldots \cup DB_N$を全体DBとよぶ.

問題例

DB_1	
1	a b
2	a b
3	a c

DB_2	
4	a
5	a
6	b c

DB_3	
7	a b c
8	a b c
9	a b c

minsup=50%

全体DBの頻出飽和アイテム集合を全て発見せよ.

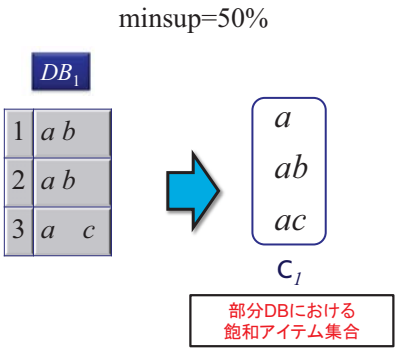
飽和アイテム集合に注目する理由

1. アイテムの種類に対して指数オーダーとなる頻出アイテム集合の数を抑えられる. [Uno et al. 05]
2. 集合族間の二項演算子 $\hat{\otimes}$(後述)で, 部分DBの飽和アイテム集合から, 全体DBの飽和アイテム集合を求めることができる. [Lucchese et al. 06]
3. 全体DBの頻出飽和アイテム集合は, 少なくとも一つの部分DBで頻出な飽和アイテム集合の部分集合である.

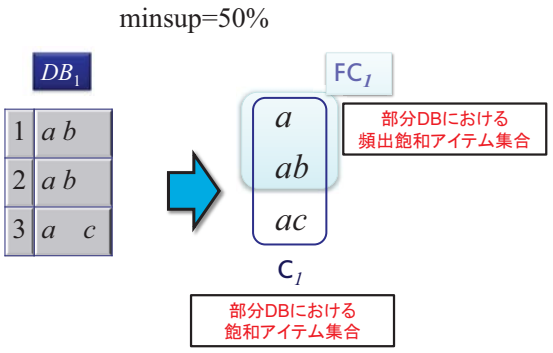
⇓

サイト間の通信路における通信コストを抑えた発見手続きを構築できることが期待される.

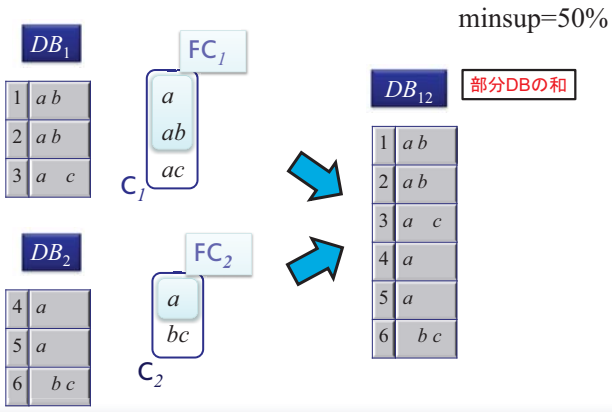
用いる記号



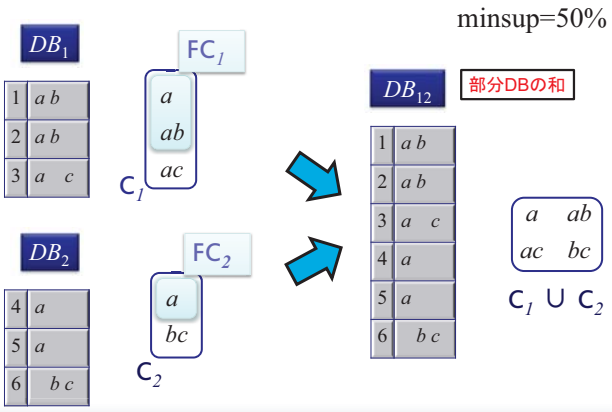
用いる記号



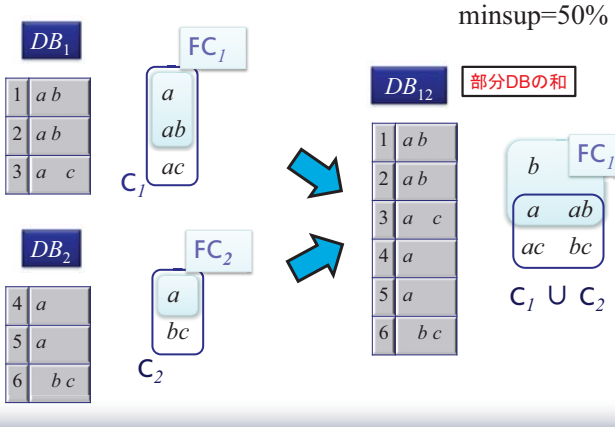
部分DBの和



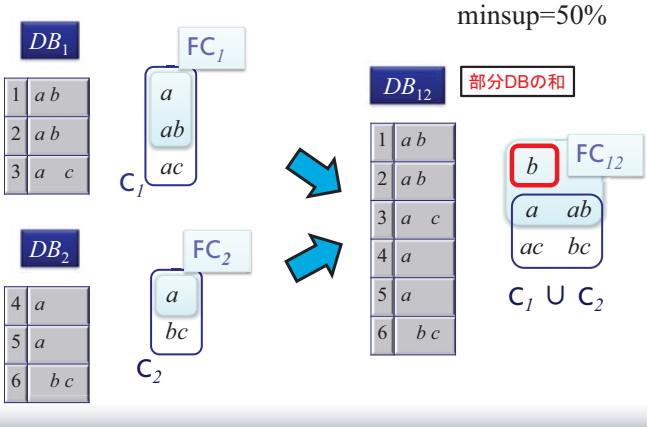
部分DBの和



部分DBの和



部分DBの和



部分DBの和における飽和アイテム集合

Iをアイテム全体の集合とする.

$S_1, S_2 \subseteq 2^I$に対して

$S_1 \otimes S_2 = \{X \cap Y \mid X \in S_1, Y \in S_2\}$

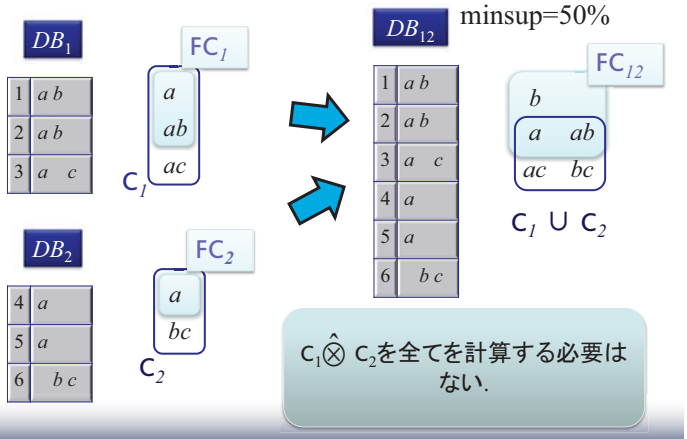
$S_1 \hat{\otimes} S_2 = S_1 \cup S_2 \cup (S_1 \otimes S_2)$ と定義する.

補題: $S_1, S_2, S_3 \subseteq 2^I$ に対して
$(S_1 \hat{\otimes} S_2) \hat{\otimes} S_3 = S_1 \hat{\otimes} (S_2 \hat{\otimes} S_3)$ が成り立つ.

$S_1 \hat{\otimes} S_2 \hat{\otimes} S_3 = (S_1 \hat{\otimes} S_2) \hat{\otimes} S_3$ とおく.

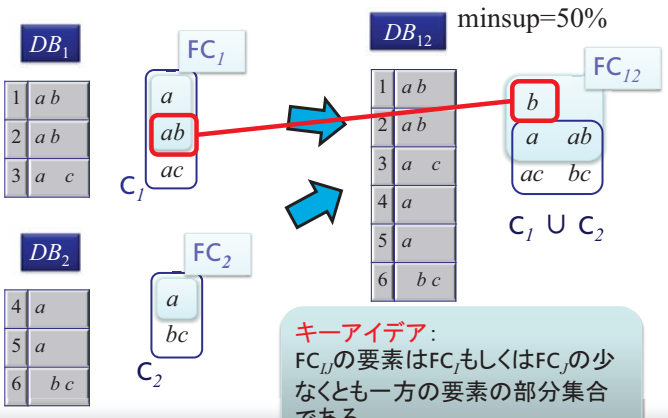
定理[Lucchese et al.]: $DB_1, \ldots, DB_N$に対して
$C_{12\ldots N} = C_1 \hat{\otimes} C_2 \hat{\otimes} \ldots \hat{\otimes} C_N$ が成り立つ.

手続きの方針



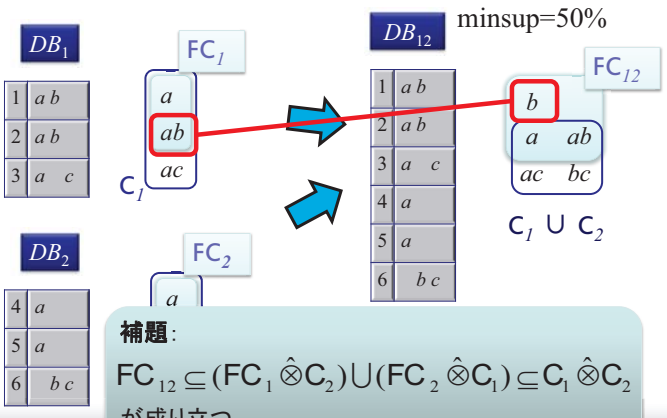
$C_1 \hat{\otimes} C_2$を全てを計算する必要はない.

手続きの方針



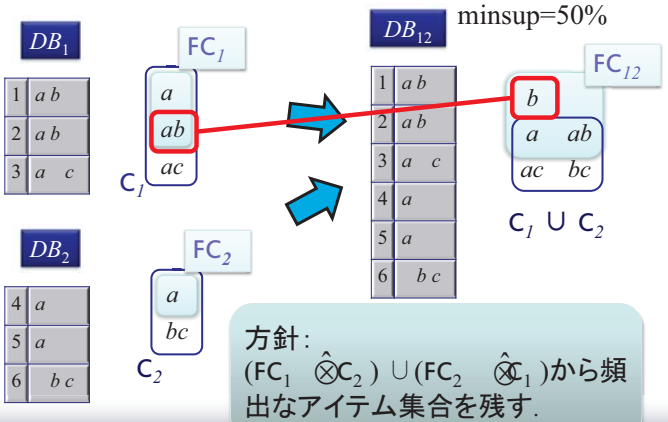
キーアイデア:
FC_{IJ}の要素はFC_IもしくはFC_Jの少なくとも一方の要素の部分集合である.

手続きの方針



補題:
$FC_{12} \subseteq (FC_1 \hat{\otimes} C_2) \cup (FC_2 \hat{\otimes} C_1) \subseteq C_1 \hat{\otimes} C_2$
が成り立つ.

手続きの方針



方針:
$(FC_1 \hat{\otimes} C_2) \cup (FC_2 \hat{\otimes} C_1)$から頻出なアイテム集合を残す.

手続き FC-Combine$_0$ の概要

FC-Combine$_0$ (FC-Combineの部分手続き):

$I, J \subseteq \{1, \ldots, N\}$とする.

- 入力
  $(DB_I$から$)\ FC_I \ \bigcup_{k \in I} C_k$ $(DB_J$から$)\ FC_J, \ \bigcup_{k \in J} C_k$
- 出力
  $FC_{IJ}, \ (\bigcup_{k \in I} C_k) \cup (\bigcup_{k \in J} C_k)$
- 手続き:
  1. $S_I := \mathrm{ClosedsetSum}(FC_I, FC_J \cup (\bigcup_{k \in J} C_k))$;
  2. $S_J := \mathrm{ClosedsetSum}(FC_J, FC_I \cup (\bigcup_{k \in I} C_k))$;
  3. $FC_{IJ} := freq(S_I \cup S_J)$;
  4. return $FC_{IJ}, \ (\bigcup_{k \in I} C_k) \cup (\bigcup_{k \in J} C_k)$

手続き FC-Combine$_0$ の概要

FC-Combine$_0$（FC-Combineの部分手続き）：

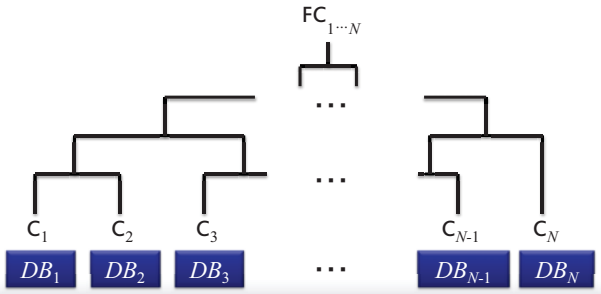
$I, J \subseteq \{1,$

- 入力
  $(D$
- 出力
  $FC_$
- 手続き
  1. S_I
  2. $S_J := \mathrm{ClosedsetSum}(FC_J, FC_I \cup (\bigcup_{k \in I} C_k))$;
  3. $FC_{IJ} := freq(S_I \cup S_J)$;
  4. return FC_{IJ}, $(\bigcup_{k \in I} C_k) \cup (\bigcup_{k \in J} C_k)$

FC-Combine:
$FC_1, \ldots, FC_N, C_1, \ldots, C_N$から
トーナメント方式で
FC-Combine$_0$を行っていき,
$FC_{1\cdots N}$を出力する手続き

トーナメント方式

トーナメント方式で手続きを行っていき, $FC_{1\cdots N}$を出力する.
(演算子の性質から, どのような順序で行ってもよい)



手続き SFC-Combine

- **SFC-Combine**: 手続きFC-Combineを暗号化を用いて行う.
- **Pohlig-Hellman encryption scheme**
  - ある大きい素数pをお互いのサイトで共有し,
    それぞれのサイトは$e \times d \equiv 1 \pmod{p-1}$となるような$e$, dの組をランダムに選ぶ.
  - 平文Mの暗号化: $E(M) = M^e \bmod p$ (秘密鍵)
  - 暗号文C の複合化: $D(C) = C^d \bmod p$
  - **補題**: $E_I(E_J(M)) = E_J(E_I(M))$ が成り立つ.

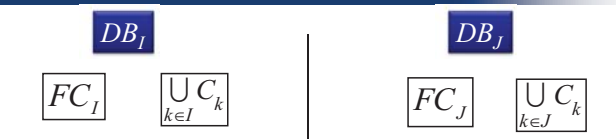
手続き SFC-Combine

- **SFC-Combine**: 手続きFC-Combineを暗号化を用いて行う.
-

SFC-Combine:
入力FC_I, FC_J, $\bigcup_{k \in I} C_k$, $\bigcup_{k \in J} C_k$内のそれぞれの
アイテムを互いの暗号鍵で暗号化し,
手続きFC-Combineを行う手続き.

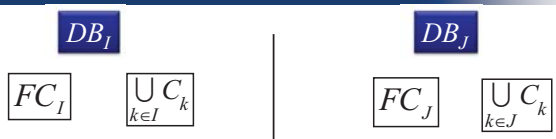
  - **補題**: $E_I(E_J(M)) = E_J(E_I(M))$ が成り立つ.

手続きSFC-Combine$_0$

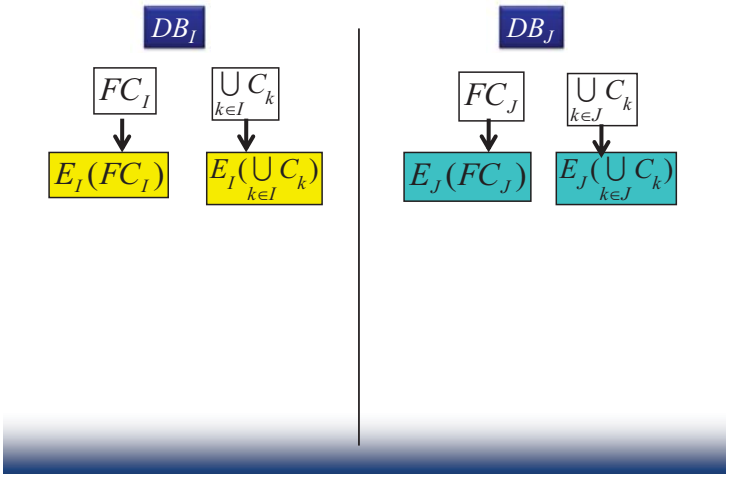


SFC-Combine$_0$:
暗号化を用いてこれらのアイテム集合の集合を
隠匿しながら, FC-Combine$_0$を行う手続き.

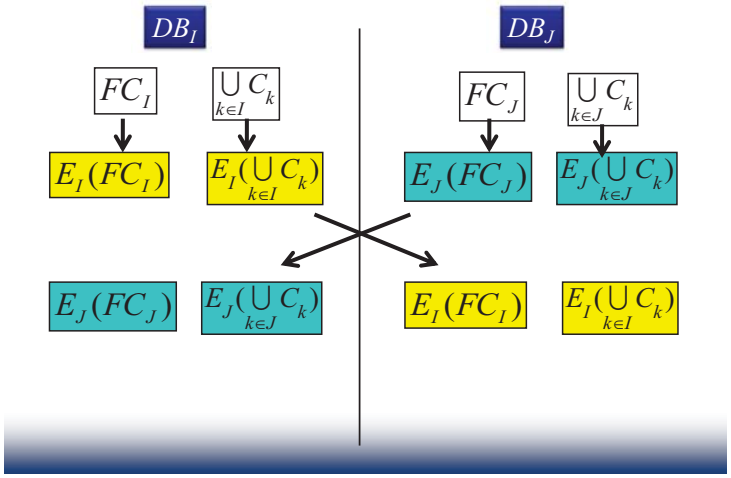
手続きSFC-Combine$_0$



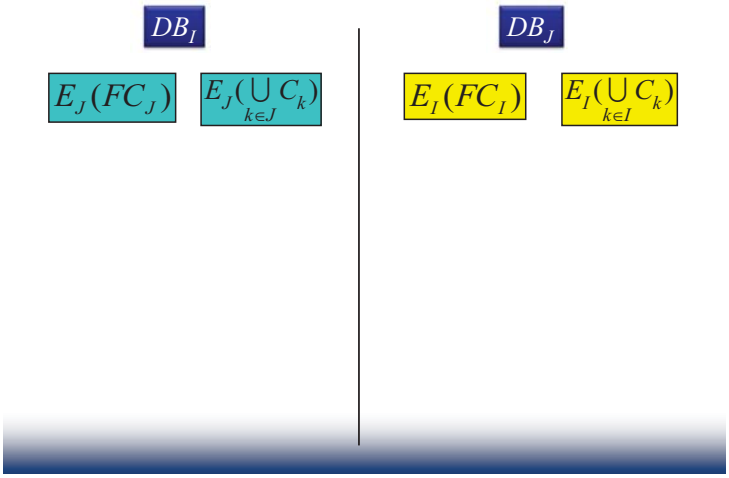
手続きSFC-Combine$_0$



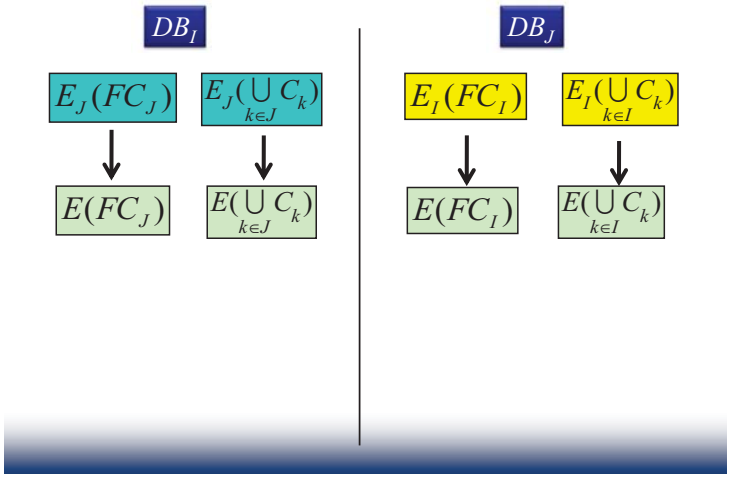
手続きSFC-Combine$_0$



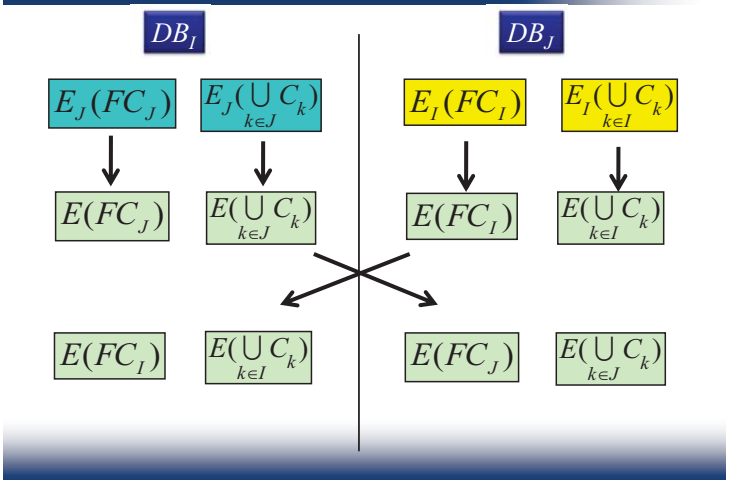
手続きSFC-Combine$_0$



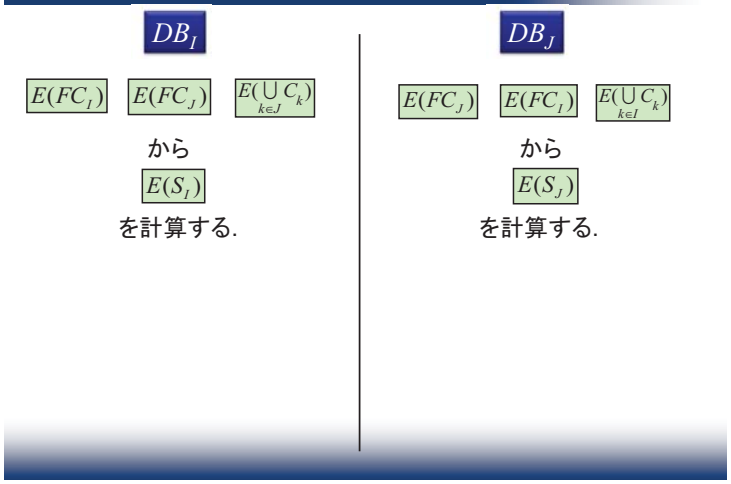
手続きSFC-Combine$_0$



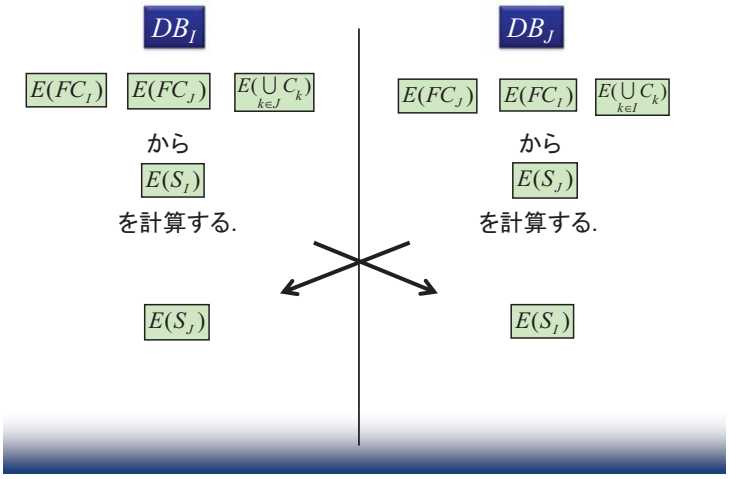
手続きSFC-Combine$_0$



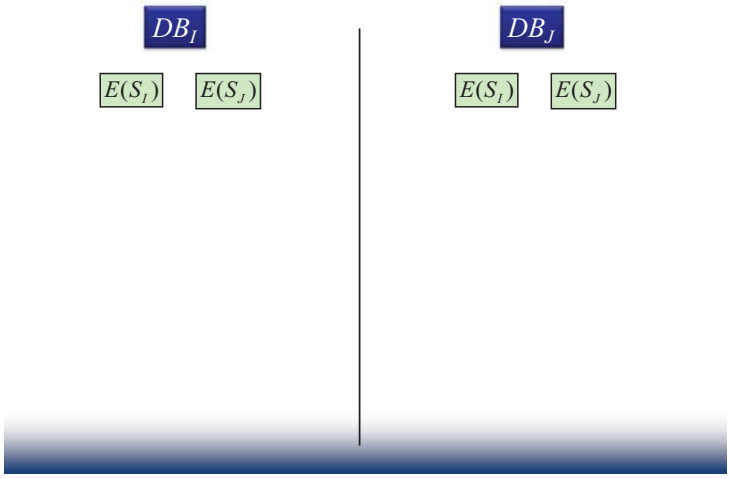
手続きSFC-Combine$_0$



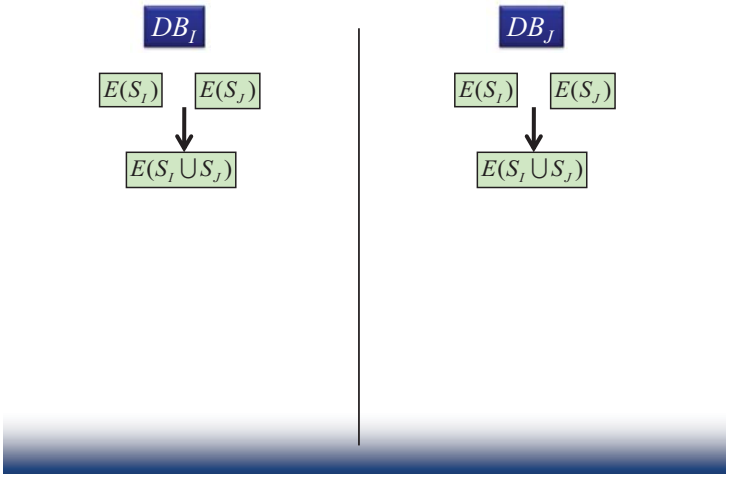
手続きSFC-Combine$_0$



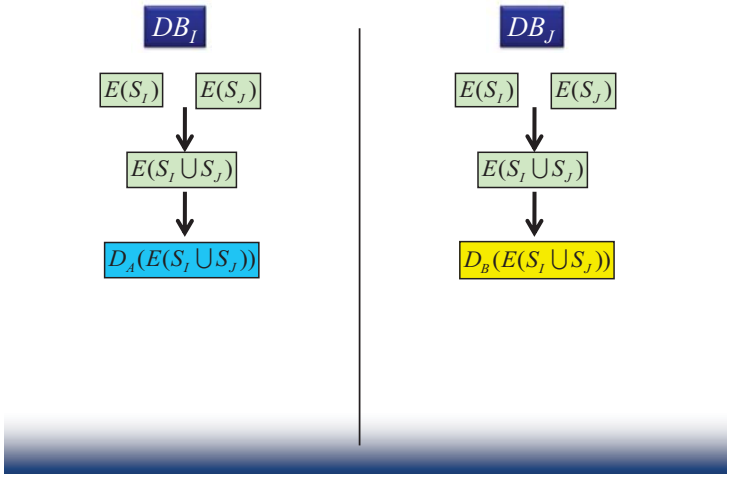
手続きSFC-Combine$_0$



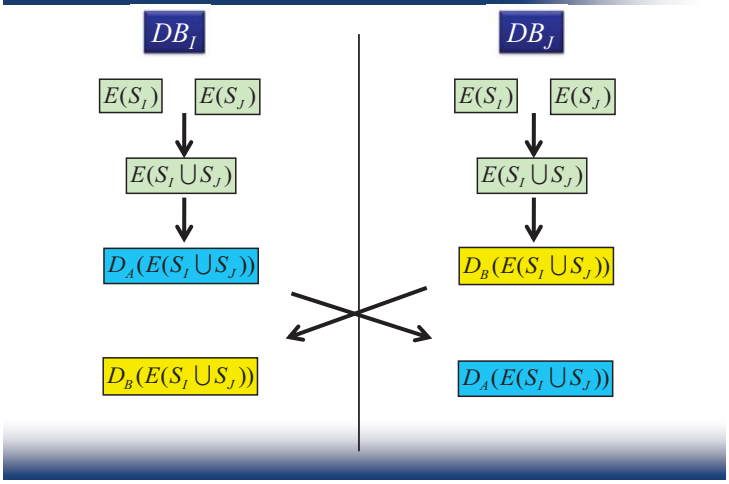
手続きSFC-Combine$_0$



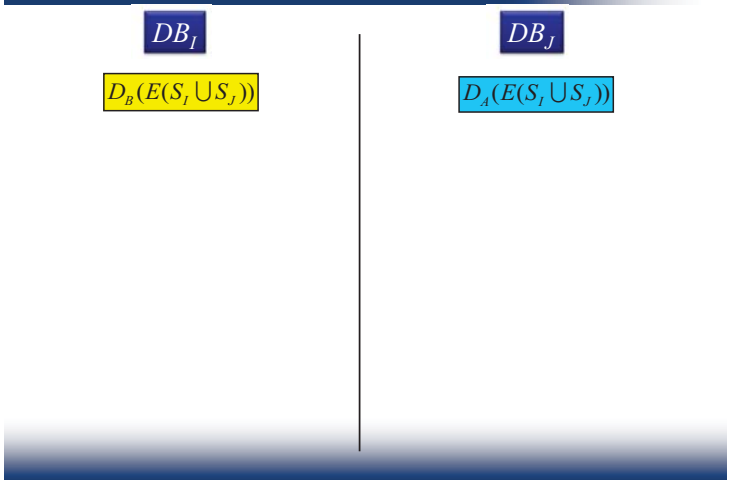
手続きSFC-Combine$_0$



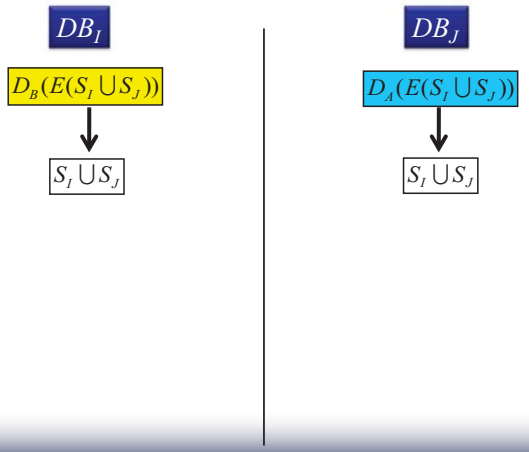
手続きSFC-Combine$_0$



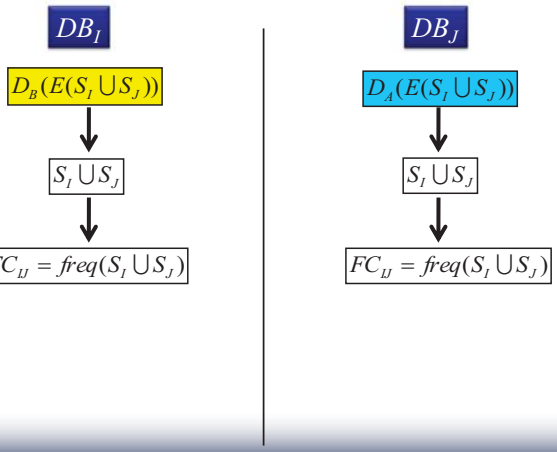
手続きSFC-Combine$_0$



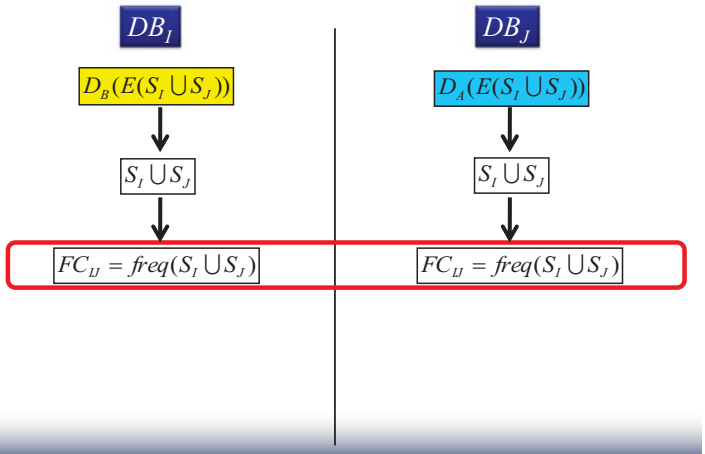
手続きSFC-Combine$_0$



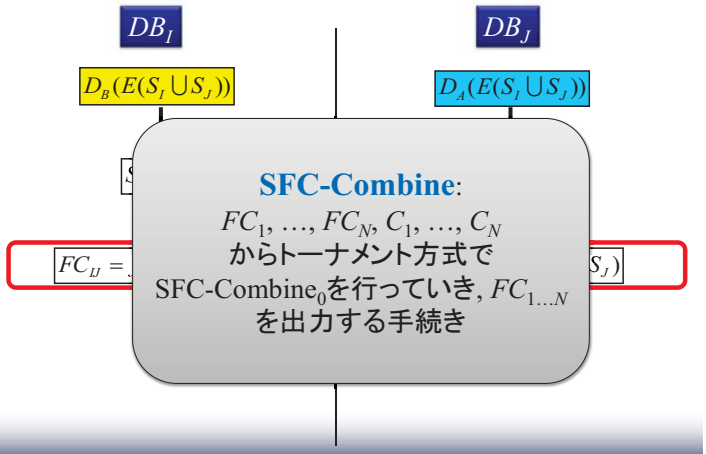
手続きSFC-Combine$_0$



手続きSFC-Combine$_0$



手続きSFC-Combine$_0$



安全性

それぞれのDBがプロトコルで決められた振る舞いから逸脱しないことを仮定する.

出力以外に漏れる情報(演算時):

SFC-Combine	(1) $C_1 \otimes C_2$ の演算時, 任意の $X \in C_1, Y \in C_2$ に対する $\lvert X \cap Y \rvert$ の大きさ (2) $C_1 \cup C_2$ の演算時, $C_1 \cap C_2$の大きさ
SFDM	

安全性の比較

	SFC-Combine (本研究)	SFDM (従来研究)	マルチパーティ・プロトコル
他サイトと共通しないアイテム	○(隠匿できる)	○	○
他サイトと共通するアイテム		○	○
他サイトと共通しないアイテム集合	○	○	○
他サイトと共通するアイテム集合			○

SFC-Combineがアイテムごとに暗号化を行っているのに対して,
SFDMはアイテム集合ごとに暗号化を行っているため

性能評価

- 以下の3つの手法に対して, 通信コストおよび安全性について比較
  - 本提案SFC-Combine
  - 分散DBから暗号化を用いて頻出アイテム集合を求める手続きSFDM [Kantarcioglu et al. 03]
  - マルチパーティ・プロトコルを用いて頻出飽和アイテム集合を求める手続き(本研究で一部を構築)
    - マルチパーティ・プロトコル[Goldreich 87]： 各参加者1, 2, …, Nが, 秘密情報$x_1, x_2, \ldots, x_N$を隠したまま, 関数$f(x_1, \ldots, x_N)$を計算するプロトコル

通信コスト(分析)

- 《分析1》全ての通信で発生するメッセージの総数
- 《分析2》同時発生する通信のメッセージを同一とみなした場合のメッセージの総数

	分析1	分析2
SFC-Combine	$10N$-10	$5\times\lceil \log_2 N \rceil$
SFDM	N 偶数の場合: $11N-12$	

1回の手続きで5回メッセージが送られており, トーナメントの高さは $\lceil \log_2 N \rceil$

通信メッセージ数 …ている.

通信コストの比較

	SFC-Combine (本研究)	SFDM (従来研究)	マルチパーティ・プロトコル
通信コスト (bit)	$O(N\cdot\lvert C_{12\ldots N}\rvert)$	$O(N^2\cdot(\sum_{k=1}^{m+1}\lvert Cand_{(k)}\rvert))$	少なくとも $O(N^2\cdot w^N)$

- m : 全体DBにおける頻出アイテム集合の大きさの最大値
- $Cand_{(k)}$: Apriorアルゴリズム[Agrawal et al. 94]を用いてつくられる, 大きさkの候補アイテム集合全体.
- w : $|C_1|, |C_2|, \ldots, |C_N|$のいずれより大きな数

通信コストの比較

	SFC-Combine (本研究)	SFDM (従来研究)	マルチパーティ・プロトコル
通信コスト (bit)	$O(N\cdot\lvert C_{12\ldots N}\rvert)$	$O(N^2\cdot(\sum_{k=1}^{m+1}\lvert Cand_{(k)}\rvert))$	少なくとも $O(N^2\cdot w^N)$

$|Cand_{(k)}|$は, ユーザが与える閾値の最小支持度が小さいとき, 爆発的に増加する性質があることが知られている[Uno et al. 03].
→最小支持度が小さいとき, 本研究は従来手法と比べて通信コストを抑えられることが期待される.

通信コストの比較(実装)

それぞれの手続きにおいて通信するアイテム集合の総数

mushroom.dat N = 4	minsup = 1%	2%	5%	10%	20%	50%
SFC-Combine	679,238	493,334	321,320	243,030	207,962	198,845
SFDM	721,325,038	177,450,024	27,312,860	4,603,279	432,581	1,240

Frequent Itemset Mining Implementations Repository : http://fimi.cs.helsinki.fi/data/
Intel Core 2 Duo 2.33GHz, 2GB RAM, Windows Vista Business SP1

minsupが小さいとき, 本提案は通信コストが小さい
→多くの興味深いパターンを小さい通信コストで発見できる

通信コストの比較(実装)

それぞれの手続きにおいて通信するアイテム集合の総数

mushroom.dat N = 4	minsup = 1%	2%	5%	10%	20%	50%
SFC-Combine	679,238	493,334	321,320	243,030	207,962	198,845
SFDM	721,325,038	177,450,024	27,312,860	4,603,279	432,581	1,240

Frequent Itemset Mining Implementations Repository : http://fimi.cs.helsinki.fi/data/
Intel Core 2 Duo 2.33GHz, 2GB RAM, Windows Vista Business SP1

minsupが大きいとき, 本提案は通信コストが大きい
→暗号化のため, 手続きの最初と最後に大きなコストがかかる

安全性の比較

	SFC-Combine（本研究）	SFDM（従来研究）	マルチパーティ・プロトコル
手続きの進行	2者間の手続きによるトーナメント方式	全体並列方式	全体並列方式

2者間の手続きでは，手続き中に漏れてしまう情報の他にも，Outputからも情報が漏れてしまう可能性がある[Kantarcioglu et al. 04].

→例えば，ある $X \in \mathsf{FC}_{IJ}$ に対し，ある $i \in X$ がサイト I で頻出でない場合，i がサイト J で頻出であることを，サイト I は知ることができる.

まとめ・今後の課題

- まとめ
  - 分散データベースから暗号化手法を用いて頻出飽和アイテム集合を全て発見する手続きSFC-Combineを構築した
  - 従来の手法より通信コストを抑えられていることを確認した
- 今後の課題
  - 複数のアイテムをまとめた上での暗号化
    - 集合演算 $\hat{\otimes}$ に対応できるような暗号化が必要
  - 全体並列方式により進行する手続きの構築