【ICDE2013勉強会】

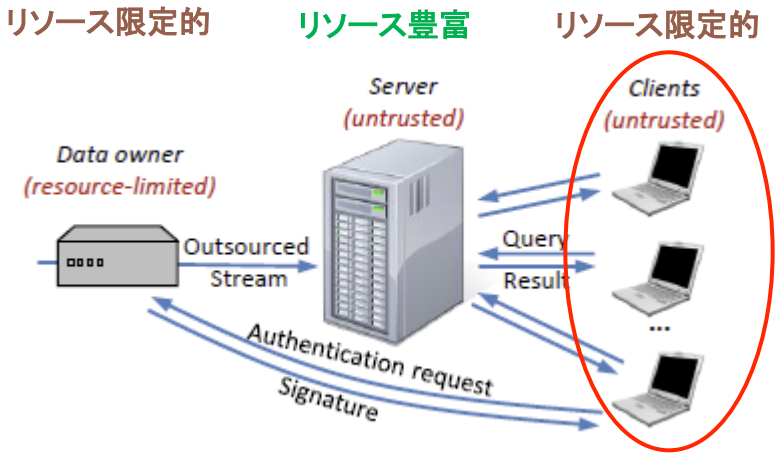
Research Session 15: Data Trust

担当：梅原，澤野，宮崎（京大）

Publicly Verifiable Grouped Aggregation Queries on Outsourced Data Streams

Suman Nath, Ramarathnam Venkatesan
Microsoft Research, Redmond, USA

背景



既存手法はクライアントが信頼出来ないことを仮定できないない、または、グループ内での集約操作をサポートしていない

ストリームデータの量が増えており、
ストリームデータの集約を外部へ委託することは、
コストダウンの観点からニーズが高まっている

例) Microsoft Bing検索エンジンのクリックデータの例では10^12の異なるグループがあり、3TBのメモリが必要になる

信頼出来ないサーバー、クライアントがいる元で
集約操作をサーバーにさせたい

クエリの例)

オンラインのマーケットプレイスがあり、それぞれの商品にレイティングが与えられている
販売者は商品グループの平均の評価を知りたい
(SQLのGroup Byに相当)

サーバー、クライアントは信用できない
サーバー: ソフトウェアにバグがあるか、または、金銭的理由から嘘の操作を行うかもしれない
クライアント: サーバーと結託して、嘘の操作を行うかもしれない

手法の基本アイデア

オーナー　ランダムに生成された秘密の番号φi を各グループiに対して生成
また、それに基づき次の形でTを維持する

φ サイズ(n)
T サイズ(1)

1) Initialize: $\mathcal{T}^0 \leftarrow 1$.
2) On arrival of the τ'th tuple (a, b), set $\mathcal{T}^\tau \leftarrow T^{\tau-1} \times g^{\varphi_a \cdot b}$.

Thus, for any τ, $\mathcal{T}^\tau = \prod_{i=0}^{n-1} g^{\varphi_i \cdot w_i^\tau}$.

サーバー　合計値のベクトルrを維持

r サイズ(n)

1) Initialize: $r_l^0 \leftarrow 0, 0 \leq l < n$.
2) On arrival of the τ'th tuple (a, b) with group a, set $r_a^\tau \leftarrow r_a^{\tau-1} + b$. Also, set $r_i^\tau \leftarrow r_i^{\tau-1}$ for all groups $i \neq a$.

クライアント　g^φiとTをオーナーからrをサーバーから受け取り検証する

r サイズ(n)
g^φi サイズ(n)
T サイズ(1)

1) $\mathcal{C}$ retrieves the result vector $\mathbf{r}^\tau$ from $\mathcal{S}$.[2]
2) $\mathcal{C}$ then retrieves $\mathcal{T}^\tau, g^{\varphi_i}, 0 \leq i \leq n-1$ from $\mathcal{O}$ and computes $\mathbb{T}^\tau = \prod_{i=0}^{n-1} (g^{\varphi_i})^{r_i^\tau}$.
3) Finally, $\mathcal{C}$ accepts $\mathbf{r}^\tau$ as correct only if $\mathcal{T}^\tau = \mathbb{T}^\tau$.

問題: そもそもφの大きさがrと同じnであり、φを維持できるならrを維持できてしまう

最適化された手法

φを小さくするアイデア

ランダムなn個の数値の代わりにlog(n)個のパラメーターで構成されるランダムな数値を生成する関数を利用する

k=log2(n)として、ρ0,ρ1,…,ρk-1の数値をランダムに生成する
aのバイナリ表現をa0,a1,…,ak-1とする

$$\beta(a) = \sum_{i=0}^{k-1} a_i \rho_i \qquad \alpha(a) = g^{A\beta(a)+B}$$

先ほどのバージョンにおけるAβ(a)+Bがφaの代わりになる

Discrete Log Problemの困難さより安全性を担保

素数pを法とする剰余環Zに含まれるg,hを考えた時、logg(h)=xなる整数xがあり、g^x=hを満たすとする。この時、xを求める線形時間アルゴリズムで知られているものは存在しない。

手法の基本アイデアでのオーナーからクライアントへのφの転送にも利用されている

最適化された手法

オーナー　関数αを利用し次の形でアップデート

ρ サイズ(log2(n))
T サイズ(1)

1) Initialize: $\mathcal{T}^0 \leftarrow 1$
2) On arrival of the τ'th tuple (a, b), set $\mathcal{T}^\tau \leftarrow T^{\tau-1} \times \alpha(a)^b$

Thus, for any τ, $\mathcal{T}^\tau = \prod_{i=0}^{n-1} \alpha(i)^{w_i^\tau}$.

サーバー　合計値のベクトルrを維持(同様)

r サイズ(n)

クライアント　g^φiとTをオーナーからrをサーバーから受け取り検証する

g^(Aρ) サイズ(log2(n))
r サイズ(n)
g^B サイズ(1)
T サイズ(1)

1) $\mathcal{C}$ retrieves the result vector $\mathbf{r}^\tau$ from $\mathcal{S}$.[2]
2) $\mathcal{C}$ then retrieves $\mathcal{T}^\tau, g^{\varphi_i}, 0 \leq i \leq n-1$ from $\mathcal{O}$ and computes $\mathbb{T}^\tau = \prod_{i=0}^{n-1} (g^{\varphi_i})^{r_i^\tau}$.
3) Finally, $\mathcal{C}$ accepts $\mathbf{r}^\tau$ as correct only if $\mathcal{T}^\tau = \mathbb{T}^\tau$.

パフォーマンスとまとめ

Microsoft Bing Click LogとWorld Cup Datasetにおいて、4GBのメモリのCore2Duoのマシンで次の時間で実行が可能

Data set	Update time	Verification time
Bing Click Log	$27\mu Sec$	$\approx 5Sec$
World Cup Dataset	$30\mu Sec$	$\approx 1Sec$

Fig. 4. Overhead at the Owner (update time per tuple) and at a Client (verification time per result) under two real data sets

Microsoft Bing Click Log(1000万レコード, グループは(検索語,クリックURL)のペア)
World Cup Dataset(1億レコード, グループはユーザー37万人)

まとめ

信頼出来ないクライアント、サーバーがいる状況下で集約クエリが正しく実行されたかを検証する手法を提案し、実験により実用的な時間で計算が完了することも示した。

Trustworthy Data from Untrusted Databases

Rohit Jain (Purdue University),
Sunil Prabhakar (Purdue University)

※図・表は論文より引用

研究背景とアプローチ

前提：複数のユーザーがアクセスやアップデートを行える

要件：信頼できないサーバーから信頼に値するデータを

- 信頼に値する：以下の3要件を保障したい
  - Correctness:クエリの解となる値はデータベースからきちんと
  - Completeness：解となるべきクエリの解は漏れなく返す
  - Transactional integrity：データベースは常に最新の一貫した状態を反映
- 手段

  上記、特にTransactional integrityを解決するプロトコルの提案

既存研究との差異

【既存研究におけるassumption】

- 全てのアップデートがデータ所有者によって信頼性を確かめられ、データベースサーバーに送られる

数多くのクライアントが存在→多くのトランザクション

適用不可

- データベース所有者には正しいアップデートを判別不可

【彼らの研究により】

- 複数のクライアントに対応したプロトコル

クラウド化されたデータベースに有用

彼らの貢献

- 信頼できないサーバー上で提供されたデータベースに対する*Transactional Integrity* を保障する問題の同定

- 3要件を保障する新しい認証メカニズム

(Correctness, Completeness, Transactional integrity)

- サーバーに*indemnity*(きちんと処理を行った証拠),
  データに*trustworthy provenance* を提供する方法

- オラクル上での実装デモと評価

Tools

i) One- Way Hashing

ii) Merkle Hash Trees → MB-tree(B+-tree による拡張)

iii) Digital Signiture

- F. Li, M. Hadjieftheriou, G. Kollios, and L. Reyzin, “Dynamic authenticated index structures for outsourced databases,” in SIGMOD, 2006

- **ハッシュ木**

$$\Phi(n) = h(\Phi(n_{left})||\Phi(n_{right}))$$

- 子ノードを連結させたものに一方向ハッシュを適用

  →親ノード

  - 管理者はrootの値のみを保存(→proof)
  - サーバーから送られてきた*VO*(a,b,c)を基に計算して確認

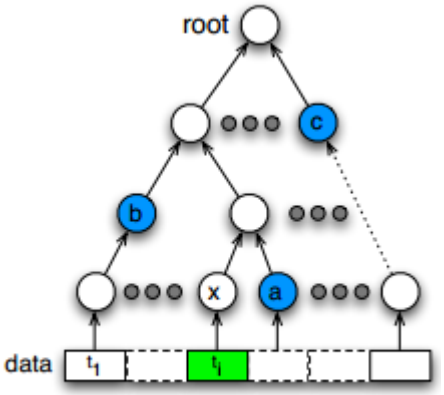
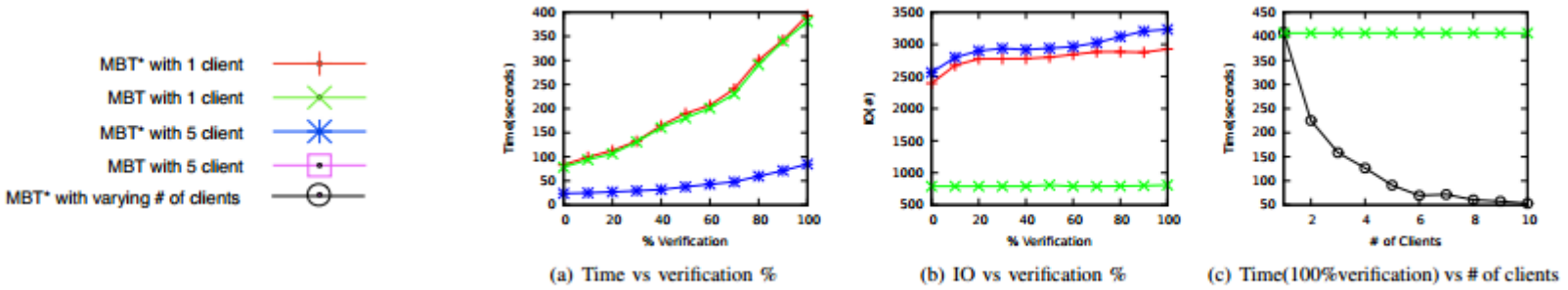


Fig. 2. An example Merkle tree

（プロトコルの詳細は割愛）

評価実験）一例

- クライアント数が同じ場合
  - 総時間は微増
  - 総IO数は増加
- クライアント数を増やしトランザクションを分配できる
  - 既存手法と違い、過去に遡ってトランザクションを認証可能
    →認証作業とトランザクションを独立に実行可



(a) Time vs verification %　(b) IO vs verification %　(c) Time(100%verification) vs # of clients

Fig. 10. Cost of Insert and verification

On the Relative Trust between Inconsistent Data and Inaccurate Constraints

George Beskales [1]
Ihab F. Ilyas [1]
Lukasz Golab [2]
Artur Galiullin [2]

[1] Qatar Computing Research Institute
[2] University of Waterloo

Introduction

対象問題： 関数従属性（FD）が必ずしも遵守されていないデータのクリーニング

FD

	GivenName	Surname	BirthDate	Gender	Phone	Income
t_1	Jack	White	5 Jan 1980	Male	923-234-4532	60k
t_2	Sam	McCarthy	19 Jul 1945	Male	989-321-4232	92k
t_3	Danielle	Blake	9 Dec 1970	Female	817-213-1211	120k
t_4	Matthew	Webb	23 Aug 1985	Male	246-481-0992	87k
t_5	Danielle	Blake	9 Dec 1970	Female	817-988-9211	100k
t_6	Hong	Li	27 Oct 1972	Female	591-977-1244	90k
t_7	Jian	Zhang	14 Apr 1990	Male	912-143-4981	55k
t_8	Ning	Wu	3 Nov 1982	Male	313-134-9241	90k
t_9	Hong	Li	8 Mar 1979	Female	498-214-5822	84k
t_{10}	Ning	Wu	8 Nov 1982	Male	323-456-3452	95k

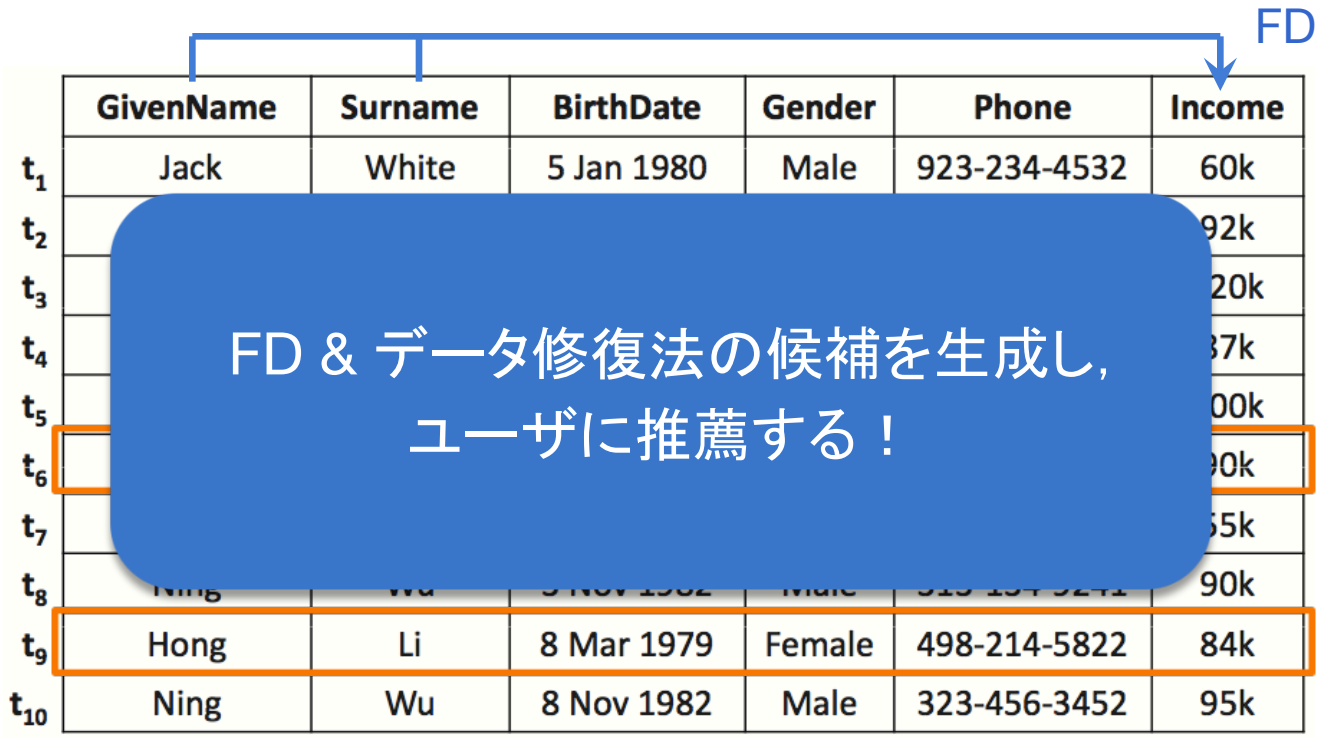
??

FDが古くなっている ⟶ データに合うようFDを修復
データに問題がある ⟶ FDに合うようデータを修復

} 信頼性をどう判断？

Introduction

対象問題： 関数従属性（FD）が必ずしも遵守されていないデータのクリーニング



FDが古くなっている ⟶ データに合うようFDを修復
データに問題がある ⟶ FDに合うようデータを修復

} 信頼性をどう判断？

Minimal Repair

Minimal Repair

FDの集合Σとタプル集合Iが与えられたとき，その修復(Σ', I')について，

$dist(\Sigma, \Sigma'), dist(I, I')$ の少なくとも一方が他の任意の修復法より小さい（Pareto-Optimal）

距離空間の例

（ for Σ ： 改変されたFDの数（追加する属性の性質で重み付け）
for I ： 改変されたセルの数

Relative Trust : τ

I'を得るために改変可能なセル数をτまでと制限し，Σの近傍で解を探索

➡ このとき発見される minimal repair を τ-constraint repair と呼び，
τを様々に変化させると全ての minimal repair が発見できる

Repair Generating Algorithm

入力: FD集合 $\sum$, タプル集合 I

Relative trust τ に対し,

（1） FD集合$\sum$の修正候補集合 $S(\sum)$ から, $\sum'$を選ぶ

ただし, I を τ セル以下の修正で $\sum'$を満たす I' に書換え可能なもの.

➡ 修正操作のコスト推定が必要だが, NP-Hard なので近似！

かつ, $dist(\sum, \sum')$ が極小となるもの.

（2） $\sum'$を遵守するよう I を極小回修正して I'を得る

これを $\tau \in [\tau_l, \tau_u]$ に対して実行し, 得た Minimal repair $(\sum', I')$ を Relative trust τ のレベル別に出力

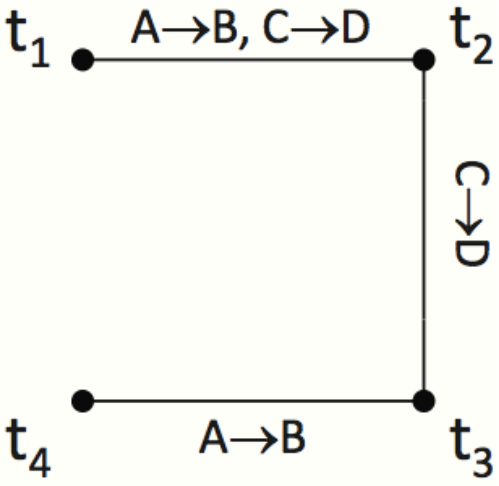
Repair Generating Algorithm

（1） FD集合$\sum$の修正候補集合 $S(\sum)$ から，$\sum'$を選ぶ

FDを破っているタプルの組を，無向グラフで表現

	A	B	C	D
t_1	1	1	1	1
t_2	1	2	1	3
t_3	2	2	1	1
t_4	2	3	4	3

Σ= { A→B, C→D }

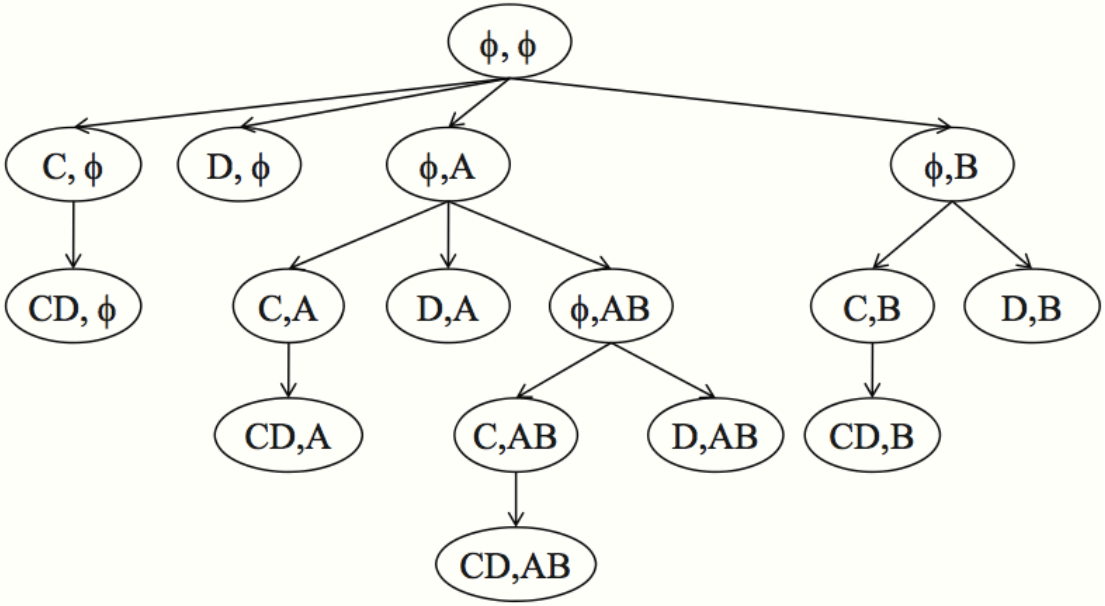


各組に関して，Difference Set を抽出 （例：t_1,t_2 に対して B,D）

Repair Generating Algorithm

（1） FD集合$\sum$の修正候補集合 $S(\sum)$ から，$\sum'$を選ぶ

FDの修正（左辺に属性を追加）パターンを木で表現



A*アルゴリズムで最小コストの枝を探索

Repair Generating Algorithm

（2） Σ'を遵守するよう I を極小回修正して I'を得る

	A	B	C	D
t_1	1	1	1	1
t_2	1	2	1	3
t_3	2	2	1	1
t_4	2	3	4	3

$\Sigma' = \{CA \rightarrow B, C \rightarrow D\}$

$C_{2opt} = \{t_2\}$

t_2 を修正する必要有!!

Fixed Attributes

t_2		2			$t_c = (v_1^A, 2, v_1^C, v_1^D)$:B を追加 ○
t_2		2	1		$t_c = (v_1^A, 2, 1, 1)$:C を追加 ○
t_2	v_1^A	2	1		$t_c = (v_1^A, 2, 1, 1)$:A を追加 △
t_2	v_1^A	2	1	1	$t_c = (v_1^A, 2, 1, 1)$:D を追加 ×

（修正!!）

※ 属性の挿入順序はランダム

Evaluation

$\tau = 0$ としたとき，FDを適切に修正できた

		FD Error	Data Error	FD Precision	FD Recall	Data Precision	Data Recall	Combined F-Score
既存	Uniform-Cost Repairing	80%	0%	1	0	0	1	0
		50%	5%	1	0	0.09	0.71	0.08
		30%	5%	1	0	0.04	0.70	0.04
		0%	5%	1	1	0.03	0.69	0.53
提案	Relative-Trust Repairing	80%	0%	0.5	0.4	1	1	0.72
		50%	5%	0.33	1	0.06	0.01	0.26
		30%	5%	0.03	0.01	0.03	0.01	0.26
		0%	5%	0.13	0.13	0.13	0.13	0.56

Fig. 9. The maximum quality achievable by our algorithm and the algorithm in [6]

悪く見えるが，様々なτを考慮しているため．

適切なτを設定した場合には，良い性能が出てます！