# プライベート問合せにおける 問合せ頻度を用いた制約緩和手法

川本 淳平[1,a)]　Patricia L. Gillett[2]　佐久間 淳[1,3]

**概要**：本論文では，**プライベート問合せ**における問合せの頻度を用いた制約緩和手法を提案する．プライベートな問合せとは，データベースの利用者が何を問い合わせているのか，すなわち**検索意図**を隠したまま目的のデータを取得することを可能とする問合せ手法のことである．既存手法の多くは，次の二つを用いてプライベートな問合せを実現している．i) 問合せをサーバが実際の値を知ることはできないが問合せ処理のみ実行できる形式に符号化する．ii) サーバは問合せ処理時にサーバが保持するすべてのタプルを走査する．これらにより，データベースサーバが攻撃者となる場合であっても，どのタプルが実際に問い合わせられたのかを隠すことができる．しかし，この二つ目の条件によって，サーバにおける問合せ処理コストは，サーバが保持するタプルの総数を $n$ として $O(n)$ となる．本論文で提案する制約緩和手法は，この二つ目の条件を緩和し多くの場合でデータベースの一部のみの走査でプライベートな問合せを可能とする．提案手法は，一次元データベースに対する一致問合せのみならず，範囲問合せや二次元データベースに対する一致問合せにも用いることができる．

## 1. はじめに

プライベート問合せは利用者が何を問い合わせているのか（**検索意図**）をサーバに隠したまま目的のデータを取得する仕組みであり，サービスとして提供されているデータベース (DaaS; Database as a Service) を安全に利用するために用いられている．利用者の検索意図は商業的に重要な情報である一方，利用者のプライバシ情報を含んでいる可能性があるからである．例えば，データベースサービスを利用したアプリケーションとしてニュースサイトを考える．攻撃者が，利用者がどんな記事を問合せ閲覧していたのかを取得することで，その利用者の趣味や政治観，宗教観などを推測することが可能となる．そして，これらは利用者によっては秘密にしておきたい種類の情報である場合が少なくない．また，より深刻な場合として，薬品データベースの例が考えられる．利用者は，病院から処方された薬に関する情報を得るために，薬品データベースを利用するとする．このとき，利用者の検索意図，すなわちどんな薬品名を問い合わせたのかという情報から，その利用者の病名を推定することが可能である．こうしたリスクを回避するためにプライベート問合せは用いられる．

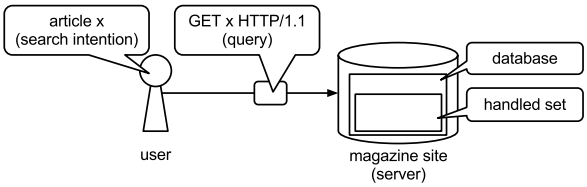



**図 1**　検索意図，問合せ及び走査集合．
**Fig. 1**　Search intention, query, and handled set.

本論文では，検索意図，問合せそして走査集合という三つの概念を用いてプライベート問合せの性質を議論する．**検索意図** *(search intention)* とは，利用者が実際に何をサーバに問い合わせたのかそのものを言う．**問合せ** *(query)* とは，検索意図に合致する情報をサーバから取得するために利用者が送信するデータのことを言う．この問合せは，一般的にサーバと利用者が合意しているプロトコルで定義された形式で書かれている．**走査集合** *(handled set)* とは，サーバが受け取った問合せを処理し問合せ結果を作成するために走査しなければならない最小のデータ集合を言う．最小のデータ集合であるので，サーバが取り得る最善の戦略で問合せを評価した場合においても操作せざるを得ないデータ集合が走査集合である．**図 1** は，あるニュースサイトの例における三つの概念を図示したものである．あるニュースサイト (magazine site) の利用者 (user) が，記事

[1] 筑波大学大学院システム情報工学研究科
Faculty of Engineering, Information and Systems, University of Tsukuba
[2] École Polytechnique de Montréal
[3] 科学技術振興機構さきがけ
Japan Science and Technology Agency
[a)] junpei@mdl.cs.tsukuba.ac.jp

$x$ を閲覧したいと考えているとする．従って，この利用者の検索意図は $x$ になる．今，このニュースサイトを利用するために簡略化した HTTP プロトコルを用いるとすると，この利用者は記事 $x$ を取得するために問合せ “GET x HTTP/1.1” をサーバに送る．このニュースサイトのサーバは問合せを受け取り，サーバが持っているデータベースの一部を走査し記事 $x$ を発見する．そして，利用者へ問合せ結果として記事 $x$ を送信する．この問合せ処理におけるサーバが走査する必要のあるデータ集合は，データベースの設計と使用プロトコルに依存する．例えば，データベースが $x$ の探索用に効率的なハッシュ索引を用意可能な場合，$x$ のみを走査し問合せ結果を作成できるであろう．しかし，一切の索引が利用できずアイテム間に順序も定義されていない場合は全てのアイテムを走査しなければならない．このニュースサイト場合サーバが効率的な索引を用意可能と仮定すると，走査集合の定義より検索意図 $x$ に対する走査集合は $\{x\}$ となる．

多くのプロトコル [1], [2] は，次の事柄によってプライベート問合せを実現している．i) サーバが問合せ処理を実行することはできるが解読することができないような符号化を問合せに施す．ii) 走査集合が常にデータベース全体となることを強制する．すなわち，サーバはどのような方法を用いて問合せ処理を効率化しようとも，データベース全体の走査が必要となる．一つ目の条件を実現するために，既存のプロトコルは暗号化などの技術を用いている．そして，二つ目の条件すなわちサーバに課された制約により，既存のプロトコルはデータベースサーバに対してもデータベース全体うちどのデータが実際に求められていたのかを隠している．一方，この制約によって，データベース内の総アイテム数を $n$ として，サーバは $O(n)$ の計算コストをあらゆる問合せに対しても負うことになる．これは，既存のプロトコルを巨大なデータベースに適用することが難しいということを表している．

一方で，我々は利用者の検索意図をデータベース全体の中から隠さなくても良い場合があると考えている．例えば，先ほどのニュースサイトの例では，利用者がすべての記事の中からどの記事を閲覧したのかまでを隠さなくとも，一部の記事の中からどの記事を閲覧したのかを隠すだけでもその利用者の政治観や宗教観を秘匿することが考えられる．それゆえ，このような場合では先ほどの制約を緩和しデータベースの部分集合から検索意図を隠すことができる．bbPIR [3] は，この制約緩和をナイーブに実現したプロトコルであり，検索意図 $x$ をデータベース全体から隠すのではなく記事番号が $x$ に近い $k-1$ 個のアイテムの中から隠す．すなわち，bbPIR における走査集合は $x$ を含む $k$ 個のアイテムとなる．しかし，このナイーブな方法は安全ではない．例えば，ニュースサイトの例では，利用者の検索意図である記事 $x$ が人気記事で，その他の $k-1$ 記事があまり問い合わせられていないことを攻撃者が知っていたとする．このとき，利用者の検索意図が $x$ であることは高確率で推定されてしまう．

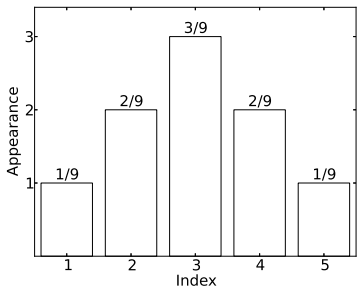



**図 2** 検索意図の頻度に関する例.
**Fig. 2** A small example of frequency.

そこで，本論文では，検索意図の頻度をもとにした安全な制約緩和手法を提案する．提案手法の基になるアイデアは，検索意図が人気のアイテムである場合，検索意図は高い確率で推定されうるため，多くのアイテムの中から隠す必要がある．一方で，検索意図があまり人気でないアイテムの場合，それほど大きなアイテム集合の中から隠さなくとも検索意図が推定される確率は大きくないというものである．制約緩和を行わないプロトコルを **完全なプロトコル** と呼ぶことにすると，提案手法は上記のアイデアを基に多くの場合で走査集合を完全なプロトコルに比べて削減する．しかし，同時に安全性についても確保しており，提案手法では制約緩和によるリスクが完全なプロトコルにおける最大のリスクを超えないことを保証する．一般的に，完全なプロトコルは安全であると考えられているため，その最大リスクを超えない本提案手法も安全であると言える．本論文では，一次元データベースにおける一致検索に対する制約緩和手法を導入した後，範囲問合せに対する制約緩和手法と二次元データベースに対する一致検索に拡張する．それゆえ，提案手法は様々な種類のデータベースに適用することが可能である．

本論文で提案する検索意図の頻度分布を用いた制約緩和手法の概要は次の通りである．例として 5 個のアイテムからなり，利用者の問合せ回数及び検索意図の頻度が **図 2** のようなデータベースを考える．完全なプロトコルでは，検索意図が 1 の時でも 3 の時でも走査集合は $[1,5]$ である．しかし，図 2 に示す検索意図の頻度を攻撃者が知っている場合，走査集合が $[1,5]$ であっても，検索意図が 1 及び 3 と推測できる確率，すなわち問合せリスクはそれぞれ $1/9$ と $3/9$ と考えることができる．本論文で提案する緩和手法は，いずれの検索意図に対してもリスクが $3/9$ 以下という条件の下，走査集合を最小化する．この例では，検索意図が 1 の場合は $[1,2]$ が緩和された走査集合になる．このときのリスクは $1/(1+2) = 1/3$ である．また，検索意図が 3 の場合は緩和されず走査集合は，$[1,5]$ のままである．

## 2. 基本事項

本論文では，データベース $D$ は $n$ 個のアイテム $\{t_1, t_2, \cdots, t_n\}$ からなり，利用者の目的は $x$ 番目のアイテム $t_x$ を求めることであるとする[*1]．この設定において，利用者の検索意図は $x$ である．例えば，1 節で用いたニュースサイトの場合，このサイトが $n$ 本の記事を掲載しており利用者に記事番号と見出しの組からなる目次を提供しているとする．読者，すなわちこのサイトの利用者は，目次から興味ある見出しを選びその記事番号 $x$ をサーバに問い合わせる．薬品データベースの場合，同様にこのデータベースは $n$ 個の薬品情報を持っており，薬品名と薬品 ID からなる目次を提供しているとする．利用者は目次から薬品 ID を取得し，その薬品の詳細をデータベースに問い合わせるといった場合を考えることができる．

cPIR [1] や IPP 法 [2] などといった既存のプライベート問合せを実現するプロトコルでは，平文の検索意図 $x$ はサーバに直接には送信されず符号化された問合せがサーバに送られる．サーバは受け取った問合せを基にデータベース $D$ の中のいくつかのデータに対して演算を行い問合せ結果を計算する．サーバが検索意図 $x$ に対する問合せ結果の計算に用いなければならない最小のデータ集合，すなわち走査集合を $H(x)$ と書くと，$H(x) \subseteq D$ を満たす．例えば，cPIR ではサーバはすべてのデータを用いて問合せ結果を作成するため $\forall x \in D : H_{cPIR}(x) = D$ である．また，ナイーブな制約緩和手法である bbPIR [3] では，$k$ 個のアイテムからなる部分データベースに対して問合せを行うため，$\forall x \in D : |H_{bbPIR}(x)| = k$ である．

次に，検索意図 $x \in D$ の頻度を $\mathrm{Freq}(x)$ と書く．本論文では，この頻度が正規化され $\sum_{x \in D} \mathrm{Freq}(x) = 1$ を満足すると仮定する．この検索意図の頻度は，サーバ，利用者及び攻撃者が知り得る公開情報であるとする．この頻度を用いて，プライベート問合せにおける **問合せリスク** を次のように定義する．

**定義 2.1** サーバは，検索意図が $x$ である問合せに対して走査集合 $H(x)$ に含まれるアイテムを走査し問合せ結果を作成する．よって，この検索意図 $x$ に対する問合せリスクを $H(x)$ から $x$ を推定できる確率として定義する．つまり，問合せリスク $\mathrm{Risk}(x|H(x))$ を次のように定める．

$$\mathrm{Risk}\,(x|H(x)) = \frac{\mathrm{Freq}(x)}{\sum_{y \in H(x)} \mathrm{Freq}(y)}$$

本論文では，検索意図の頻度を知っている攻撃者を想定している．攻撃者は問合せから利用者の検索意図 $x$ を知ることはできないが，サーバが問合せ処理に用いた走査集合 $\forall x \in D : H(x)$ は知ることができるとする．そして，攻撃者の目的を利用者の検索意図 $x$ を走査集合 $H(x)$ と検索意図の頻度 $\mathrm{Freq}(x)$ から推定することとする．

**定義 2.2** 我々は，任意の検索意図 $x$ に対して，その問合せリスクが完全なプロトコルにおける最大の問合せリスクより下回っているとき，制約緩和された走査集合 $H_{relaxed}$ は安全であると言う．言い換えれば，$\forall x \in D$ に対して，

$$\mathrm{Risk}\,(x|H_{relaxed}(x)) \leq \max_{y \in D} \mathrm{Risk}\,(y|H_{complete}(y)) \quad (1)$$

が満たされるとき $H_{relaxed}$ は安全であると言う．
この定義は，我々の制約緩和手法がある検索意図に対しては問合せリスクを増加させることを意味する．しかし，その問合せリスクは完全なプロトコルにおける最大のリスク以下に抑えることができる．一般に制約緩和を行わない完全なプロトコルは安全であると認められている．そのため，その最大リスクを超えなければ制約緩和を行いリスクが部分的には増大しても安全であると考えることができる．

最後に，問合せの処理コストを定義する．本論文では，検索意図 $x$ に対する対する問合せを処理するコストを $C(x)$ と書き，走査集合の大きさで評価する．すなわち，$C(x) = |H(x)|$ である．また，検索意図全体に対する問合せ処理コストの期待値を考え $C$ と書くことにする．したがって，$C = \sum_{x \in D} \mathrm{Freq}(x) \times C(x)$ と計算することができる．完全なプロトコルでは，サーバは検索意図によらず常にすべてのアイテムを走査する．つまり $\forall x \in D$: $H_{complete}(x) = D$ かつ $C_{complete}(x) = n$ であり，$C_{complete} = \sum_{x \in D} \mathrm{Freq}(x) \times C_{complete}(x) = n \sum_{x \in D} \mathrm{Freq}(x) = n$ と評価することができる．

## 3. 制約緩和

本論文で我々が提案する制約緩和手法によって，サーバは問合せ処理のためにデータベース全体を走査する必要はなくなり，その一部のみを走査する．簡単のため，まず一次元データベースにおける一致検索に対する制約緩和手法から導入する．提案手法を用いると，利用者の検索意図が $x$ の場合，走査集合は連続するアイテムの集合 $H_{relaxed}(x) = \{t_i, t_{i+1}, \cdots, t_j\}$ となる．ここで，$i \leq x \leq j$ であり，$H_{relaxed}(x)$ は $t_x$ を含む．$H_{relaxed}(x) \subseteq H_{complete}(x)$ であるから，サーバは多くの場合計算コストを削減できる．一般に制約緩和はマイナスの要素も含む．ここでは，問合せリスクの増大という形で現れる．しかし，前節までに述べているように，提案手法はこの増加した問合せリスクが完全なプロトコルにおける最大のリスクを超えないことを保証している．すなわち式 (1) を満足し，安全であることが保証されている．

上記の条件を満足する走査集合を求める問題は次のように定義できる．

**問題 3.1** 与えられた検索意図 $x$ に対して (i) $x \in H_r(x)$, (ii) $\mathrm{Risk}(x|H_r(x)) \leq \max_{y \in D} \mathrm{Risk}(y)$, (iii) $j - i$ を最小化

[*1] 範囲問合せを考える場合，利用者の目的は，ある範囲 $[x, y]$ に含まれるアイテム $\{t_x, t_{x+1}, \cdots, t_y\}$ を求めることであるとする．範囲問合せに関しては，3.1 節で詳しく議論する．

**Algorithm 1** Find $H_r(x)$.

**Require:** $D$ はデータベース，Freq は検索意図の頻度.
  $bestsofar \leftarrow |D|+1$, $best \leftarrow [0, |D|-1]$
  $bestsum \leftarrow \sum_{y \in D} \mathrm{Freq}(y)$ $i \leftarrow x$, $j \leftarrow x$
  $vsum \leftarrow \mathrm{Freq}(x)$, $\mu \leftarrow \mathrm{Freq}(x)/\max_{y \in D} \mathrm{Risk}(y)$
  **while** $vsum < threshold$ **and** $i \geq 1$ **do**
    $i \leftarrow i-1$, $vsum \leftarrow vsum + \mathrm{Freq}(i)$
  **end while**
  **if** $vsum \geq \mu$ **then**
    $bestsofar \leftarrow j-i+1$, $best \leftarrow [i, j]$, $bestsum \leftarrow vsum$
  **end if**
  **while** $i \leq x$ **do**
    **while** $vsum \geq \mu$ **do**
      $j \leftarrow j+1$, $vsum \leftarrow vsum + \mathrm{Freq}(j)$
    **end while**
    **if** $vsum \geq \mu$ **then**
      **if** $j-i+1 < bestsofar$ **then**
        $bestsofar \leftarrow j-i+1$, $best \leftarrow (i, j)$
        $bestsum \leftarrow vsum$
      **else if** $j-i+1 = bestsofar$ **and** $vsum > bestsum$ **then**
        $bestsofar \leftarrow j-i+1$, $best \leftarrow (i, j)$
        $bestsum \leftarrow vsum$
      **end if**
    **end if**
    $i \leftarrow i+1$, $vsum \leftarrow vsum - \mathrm{Freq}(i-1)$
  **end while**
  **return** $best$

**Algorithm 2** Find $H_{rr}(X)$.

**Require:** $D$ はデータベース, Freq は検索意図の頻度, $X = [x_L, x_R]$.
  $bestsofar \leftarrow |D|+1$, $best \leftarrow [0, |D|-1]$
  $bestsum \leftarrow \sum_{y \in D} \mathrm{Freq}(y)$, $i \leftarrow x_L$, $j \leftarrow x_R$
  $vsum \leftarrow \mathrm{Freq}(X)$, $\mu \leftarrow \mathrm{Freq}(X)/\max_{y \in D} \mathrm{Risk}(y)$
  **while** $vsum < threshold$ **and** $i \geq 1$ **do**
    $i \leftarrow i-1$, $vsum \leftarrow vsum + \mathrm{Freq}(i)$
  **end while**
  **if** $vsum \geq \mu$ **then**
    $bestsofar \leftarrow j-i+1$, $best \leftarrow [i, j]$, $bestsum \leftarrow vsum$
  **end if**
  **while** $i \leq x_L$ **do**
    **while** $vsum \geq \mu$ **do**
      $j \leftarrow j+1$, $vsum \leftarrow vsum + \mathrm{Freq}(j)$
    **end while**
    **if** $vsum \geq \mu$ **then**
      **if** $j-i+1 < bestsofar$ **then**
        $bestsofar \leftarrow j-i+1$, $best \leftarrow (i, j)$
        $bestsum \leftarrow vsum$
      **else if** $j-i+1 = bestsofar$ **and** $vsum > bestsum$ **then**
        $bestsofar \leftarrow j-i+1$, $best \leftarrow (i, j)$
        $bestsum \leftarrow vsum$
      **end if**
    **end if**
    $i \leftarrow i+1$, $vsum \leftarrow vsum - \mathrm{Freq}(i-1)$
  **end while**
  **return** $best$

する，(iv) 条件 iii の下で $\sum_{y \in H_r(x)} \mathrm{Freq}(y)$ を最大化する という条件を満足し連続するアイテムからなる集合 $H_r(x) = \{t_i, t_{i+1}, \cdots, t_j\} \subseteq D$ を求める．

条件 ii によって見つかった走査集合が攻撃者に対して安全であると保証しており，条件 iii によって問合せコストを最小化している．さらに，上記の二条件を満足する候補が複数存在する場合は，条件 iv によって問合せリスクを最小にする走査集合を選んでいる．

この問題 3.1 は，アルゴリズム 1 によって解くことができ，その計算量は $O(n)$ である．さらに，得られた $H(x)$ は同じに $x$ 対して再利用することができるため，$H(x)$ を予め計算して保存しておくことで問合せ時間に影響を与えること無く制約緩和が行える．また，アルゴリズム 1 によって得られた走査集合 $H_{relaxed}$ に対するサーバ計算コストの期待値は，$C_{relaxed} = \sum_{x \in D} \mathrm{Freq}(x) \times |H_{relaxed}(x)| \leq \sum_{x \in D} \mathrm{Freq}(x) \times |D| = n$ であり，完全なプロトコルにおける計算コストより小さくなる．

この制約緩和を用いた問合せは次のように行われる．サーバへ問合せを行おうとしている利用者の検索意図が $x$ であるとすると，この利用者は先ずアルゴリズム 1 を用いて走査集合 $H(x)$ を求める．次に，得られた $H(x)$ に含まれるアイテムのみからなる部分データベース上で，検索意図 $x$ を表す問合せ $q$ を作成する．そして，利用者は $H(x)$ と $q$ をサーバに送信する．サーバは，受け取った走査集合に含まれるアイテムからなる部分データベース上で問合せ $q$ を評価し結果を利用者に返却する．なお，問合せ $q$ は本制約緩和手法と共に用いるプロトコルによって定まっているものとする．例えば，cPIR に本手法を組み合わせて用いる場合は，cPIR における問合せを用いれば良い．

### 3.1 一次元データベースに対する範囲問合せへの拡張

次に，本節では先ほど提案したアルゴリズムを一次元のデータベースに対する範囲問合せへ拡張する．一致問合せと範囲問合せの差異は，検索意図が単一の値 $x$ ではなく連続する集合 $\{t_{x_L}, t_{x_L+1}, \cdots, t_{x_R}\}$ となることである．以降では，この範囲検索における検索意図を $X \subseteq D$ で表すことにする．そして，検索意図の頻度及び問合せリスクを，範囲検索における検索意図 $X$ 上に拡張する．

$$\mathrm{Freq}(X) = \frac{\sum_{x \in X} \mathrm{Freq}(x)}{\sum_{y \in 2^D} \mathrm{Freq}(y)}$$

$$\mathrm{Risk}(X|H(X)) = \frac{\mathrm{Freq}(X)}{\sum_{y \in H(X)} \mathrm{Freq}(y)}$$

また，問合せのコストとその期待値についても同様に拡張し $C(X) = |H(X)|$, $C = \sum_{X \in 2^D} \mathrm{Freq}(X) \times C(X)$ とする．

範囲検索における検索意図 $X$ に対する完全なプロトコルの走査集合は，一致検索の場合と同様にデータベース全てであり $\forall X \in 2^D : H_{complete}(X) = D$ となる．一方，提案手法では，連続するアイテム集合からなるデータベースの一部分 $H_{relaxed}(X) = \{t_i, t_{i+1}, \cdots, t_j\}$ となる．この $H_{relaxed}(X)$ を発見する問題は，先の問題 3.1 を拡張して

**Algorithm 3** Find $H_{rm}(x)$.

**Require:** $D' \in \mathbb{R}^{l\times m}$ は問合せ頻度を記録したデータベース $(D'_{i,j} = \mathrm{Freq}(i \times m + j))$.
$\mu \leftarrow D'_{e,g} / \max_{y\in D} \mathrm{Risk}(y)$, $C_{e,g} \leftarrow D'_{e,g}$
$vsum \leftarrow \sum_{i,j} D'_{i,j}$, $size \leftarrow l \times m$, $best \leftarrow (0, 0, l, m)$
$h \leftarrow 1$, $w \leftarrow 1$, $maxw \leftarrow m$, $C \leftarrow \mathrm{NegOnes}(h, w)$
$rangeC \leftarrow (e, e, g, g)$
**while** $h \le \min(l, size)$ **do**
  $M \leftarrow \mathrm{NegOnes}(h, w)$
  $M_{i,g} \leftarrow C_{i,g} \quad \forall rangeC[0] \le i \le rangeC[1]$
  $rangeM \leftarrow (rangeC[0], rangeC[1], g, g)$
  **while** $w \le maxw$ **do**
    $(i^{new}, j^{new}) \leftarrow \mathrm{argmax}(M)$, $v^{new} \leftarrow M_{i^{new},j^{new}}$
    **if** $(v^{new} > vsum)$ **or** $(v^{new} \ge \mu$ **and** $h \times w < size)$ **then**
      $vsum \leftarrow v^{new}$, $size \leftarrow h \times w$
      $best \leftarrow (i^{new}, j^{new}, h, w)$, $maxw \leftarrow \min(m, \lfloor \frac{size}{h} \rfloor)$
    **end if**
    $w \leftarrow w + 1$
    **if** $w \le maxw$ **then**
      $(C, rangeC, M, rangeM)$
        $\leftarrow \mathrm{updateM}(h, w, C, rangeC, M, rangeM, D')$
    **end if**
  **end while**
  $h \leftarrow h + 1, w \leftarrow 1$
  **if** $h \le \min(l, size)$ **then**
    $maxw \leftarrow \min(m, \lfloor \frac{size}{h} \rfloor)$
    $rangeC[2] \leftarrow \max(rangeC[2], g - maxw + 1)$
    $rangeC[3] \leftarrow \min(rangeC[3], g + maxw - 1)$
    $(C, rangeC) \leftarrow \mathrm{updateC}(h, C, rangeC, D')$
  **end if**
**end while**
**return** *best*

次のようなる.

**問題 3.2** 与えられた検索意図 $X$ に対して, (i) $X \subset H_{rr}$, (ii) $\mathrm{Risk}(X|H_{rr}(X)) \le \max_{y\in D} \mathrm{Risk}(y)$, (iii) $|H_{rr}(X)|$ つまり $j - i$ を最小化する, (iv) 条件 iii の下で $\sum_{y\in H_{rr}(X)} \mathrm{Freq}(y)$ を最大化するという条件を満足する $H_{rr} = \{t_i, t_{i+1}, \cdots, t_j\} \subseteq D$ を発見する.
この問題は, アルゴリズム 2 によって求めることができ, その計算量はアルゴリズム 1 と同様に $O(n)$ である.

### 3.2 二次元データベースに対する一致問合せへの拡張

最後に, 二次元データベースに対する一致問合せへの拡張について議論する. 本論文では, 二次元データベースを $l$ 行 $m$ 列の行列として考える. ただし, $l \times m = n$ とする. 利用者の目的は, この $l \times m$ 行列のうち $e$ 行 $g$ 列目のアイテムを取得することとする. すなわち, 利用者の検索意図は $x = (e, g)$ と書ける. 完全なプロトコルでは, 任意の検索意図 $x \in D$ に対してその走査集合 $H_{complete}(x)$ は, $D[[1, l]; [1, m]]$ である[*2]. したがって, 検索意図 $x$ に依らず $C_{complete}(x) = |H_{complete}(x)| = n$ であり, サーバは常にすべてのアイテムを走査する必要がある.

[*2] 行列 $D$ の $i_a$ 行目から $i_b$ 行目及び $j_a$ 列目から $j_b$ 列目からなる部分行列を $D[[i_a, i_b]; [j_a, j_b]]$ と書く.

今までと同様に, 提案手法ではこの制約を緩和し安全性の条件である式 (1) を満足する走査集合としてデータベースを表す行列の部分行列を求める. ある検索意図 $x = (e, g)$ に対して, 走査集合 $H_{rm}(x)$ を発見する問題は, 次のように定義できる.

**問題 3.3** 与えられた検索意図 $x = (e, g)$ に対して, (i) $x$ を含む, すなわち $top \le e < top + height$ かつ $left \le g < left + width$, (ii) $\mathrm{Risk}(x|H_{rm}(x)) \le \max_{y\in D} \mathrm{Risk}(y)$, (iii) $height \times width$ を最小化する, (iv) 条件 iii の下で $\sum_{y\in H_{rm}(x)} \mathrm{Freq}(y)$ を最大化するという条件を満足する, データベース行列 $D$ の部分行列 $H_{rm} = D[[top, top + height - 1]; [left, left + width - 1]]$ を発見する.
上記の条件を満足する最適な部分行列を求めるこの問題は, アルゴリズム 3 によって解くことができる. ここで, $\mathrm{NegOnes}(h, w)$ はすべての要素が $-1$ の $h \times w$ 行列を作成する関数である. アルゴリズムの計算量は $O(n^2)$ となるが, updateC 及び updateM と併せて詳細は付録に記す.

## 4. 評価実験

この節では, 提案手法の効果を二つの側面から評価する. 一つ目は, アイテム数 $n = 10000$ の一次元データベース $D$ に対して $x \in D$ を問い合わせた場合に, 提案手法適用後の検索意図頻度, 問合せリスク, そしてサーバ計算コストを計測する. この実験では, 元々の検索意図頻度としてガウス分布と Zipf 分布の二種類を用意した. ガウス分布のパラメータは, $\mu = n/2$ 及び $\sigma = n/6$ とし, 下記の式で定義される Zipf 分布のパラメータとして $s = 0.25$ を選んだ.

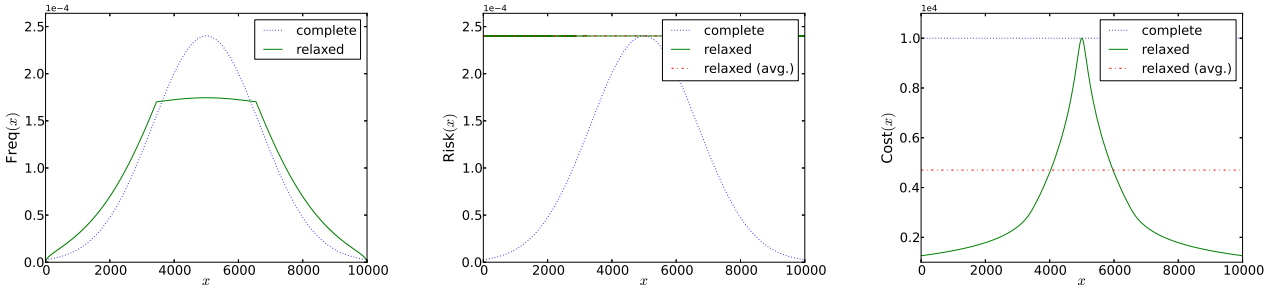
$$\mathrm{Zipf}(x; n, s) = \frac{1/x^s}{\sum_{k=1}^{n} 1/k^s}$$

Zipf 分布の場合, パラメータによって出力に大きな変化は見られなかった. そこで, グラフの変化が判別しやすいパラメータを選んだ. また, この実験ではガウス分布と Zipf 分布という二つの分布を選んだが, 一般的にどのような検索意図分布であっても, アイテム番号の並び替えにより検索意図頻度をこれらの分布に近づけることができる.

**図 3(a)** と **図 4(a)** は, それぞれガウス分布及び Zipf 分布を元々の検索意図頻度とした場合 (complete) に, 提案手法を用いた問合せから求めた検索意図の頻度, すなわちサーバが観測した見かけの検索緯度頻度 (relaxed) を示したものである. 見かけの検索意図頻度は次の式で求めた.

$$\mathrm{Freq}_{observed}(x) = \frac{\sum_{y\in D, x\in H(y)} \mathrm{Freq}(y)}{\sum_{y\in D} \mathrm{Freq}(y) \times |H(y)|}$$

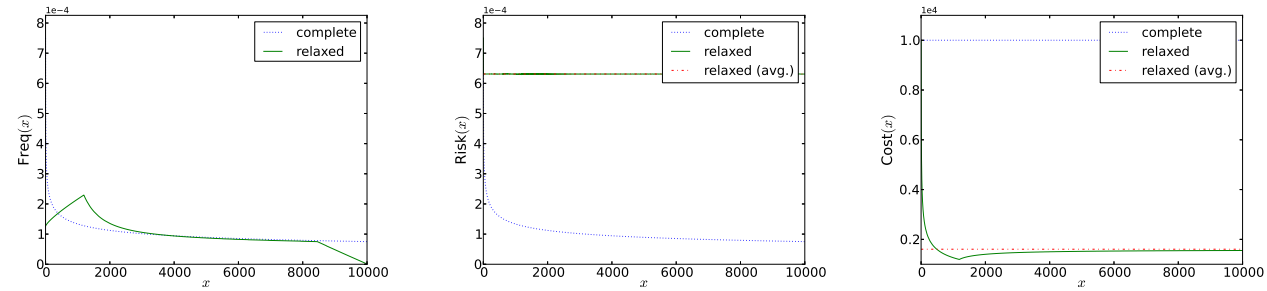
図 3(a) 及び 図 4(a) の横軸は検索意図 $x$ を表しており, 縦軸は検索意図の頻度 $\mathrm{Freq}(x)$ を表している. なお, それぞれの頻度は $\sum_{x\in D}(x) = 1$ を満たすように正規化してある. 図 3(a) 及び 図 4(a) 共に, 提案手法により頻度のピー




(a) 検索意図の頻度　(b) 問合せリスク　(c) サーバ計算コスト

**図 3** ガウス分布に従う問合せ頻度に対して観測された検索意図の頻度と問合せリスク，及びサーバ計算コスト ($n = 10000, \mu = n/2, \sigma = n/6$).

**Fig. 3** Observed frequency, query risk, and cost for a Gauss distribution ($n = 10000, \mu = n/2, \sigma = n/6$).




(a) 検索意図の頻度　(b) 問合せリスク　(c) サーバ計算コスト

**図 4** Zipf 分布に従う問合せ頻度に対して観測された検索意図の頻度と問合せリスク，及びサーバ計算コスト ($n = 10000, s = 0.25$).

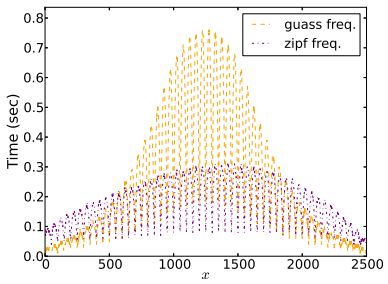
**Fig. 4** Observed frequency, query risk, and cost for a Zipf distribution ($n = 10000, s = 0.25$).

クが別の形に変わっていることが分かる．

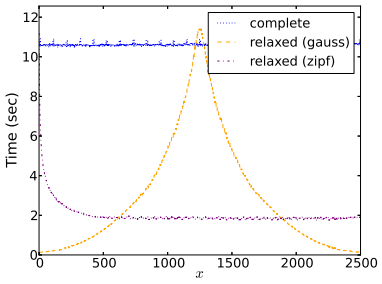
**図 3(b)** と **図 4(b)** は，横軸に検索意図 $x$ を取り，縦軸にその検索意図に対する問合せリスクを示したものである．なお，complete は完全なプロトコルにおける問合せリスクを表し，relaxed は提案手法を用いて制約を緩和した場合における問合せリスクを示している．また，$\mathrm{Risk}_{relaxed} = \sum_{x \in D} \mathrm{Freq}(x) \times \mathrm{Risk}(x|H(x))$ にて計算できる平均問合せリスクも併せて記してあるが，この平均問合せリスクと制約緩和後の問合せリスクとの間に大きな差は無かった．我々の提案手法は，完全なプロトコルにおける制約を緩和しているため，一般的に問合せリスクは完全なプロトコルに対して増加する．しかし，提案の制約緩和手法は，増加する問合せリスクが完全なプロトコルにおける最大のリスクを超えないことを保証しており，図 3(b) と図 4(b) は実際にそのことを示している．具体的に見ていくと，図 3(b) において，完全なプロトコルにおける最も高い問合せリスクは $x = 5000$ の時であり $\mathrm{Freq}_{complete}(5000)$ となる．そして，制約緩和された場合における問合せリスクを見ていくと，任意の $x$ に対してほぼ $\mathrm{Freq}_{complete}(5000)$ に等しく，また超えることは無い．図 4(b) の場合は，最も問合せリスクが高いのは $x = 0$ の時であり $\mathrm{Freq}_{complete}(0)$ となる．同様に，制約緩和された場合の問合せリスクは，任意の $x$ に対してほぼ $\mathrm{Freq}_{complete}(0)$ に等しく超えることはないことが見て取れる．

**図 3(c)** と **図 4(c)** は横軸に検索意図 $x$ を取り，縦軸にその検索意図に対するサーバの問合せ計算コストを記したものである．計算コストは完全なプロトコル (complete)，提案手法によって制約を緩和した場合 (relaxed)，そして制約緩和した場合の平均値の三つを記している．完全なプロトコルでは，サーバは常にデータベース内すべてのアイテムを走査するため，計算コストは $x$ に依存しない．一方で，制約緩和を行った場合，最も高い頻度の検索意図以外に対してはサーバにおける計算コストが削減されていることが分かる．最も高い頻度の検索意図に対しては，そのサーバ計算コストは完全なモデルと同じであるが，これは意図したとおりの動きである．

二つ目の評価実験では，実際にプライベート問合せを実現するサーバ・クライアントを用いて，問合せに要する時間を計測した．ここでは，プライベート問合せを実現する手法として，cPIR プロトコルを選び，2500 アイテムから




(a) 制約緩和された走査集合の計算時間




(b) サーバにおける問合せ処理時間

**図 5** 二次元属性に対する問合せの処理時間.
**Fig. 5** Query processing times in 2D data.

なるデータベースを構築した ($n = 25000$). また，一つ目の実験と同じように，ガウス分布に従う検索意図頻度と Zipf 分布に従う検索意図頻度の二種類を用意した．ガウス分布におけるパラメータは，先ほどと同様に，$\mu = n/2$ かつ $\sigma = n/6$ とし，Zipf 分布におけるパラメータは $s = 0.25$ とした．

**図 5(a)** は，各検索意図 $x$ に対して走査集合 $H(x)$ を求めるのに要した時間を記している．グラフはガウス分布を用いた場合 (gauss) と Zipf 分布を用いた場合 (zipf) の二種類を記してある．ここでは，cPIR プロトコルを用いているので，検索意図は $l \times m$ 行列に対して $x = (e, g)$ 番目のアイテムを求めることを意味する．グラフでは横軸に $x$ を取っているが，これは $x = e \times m + g$ によって計算したものである．図 5(a) によれば，走査集合を求めるのに要した時間は大きく波打っている．現在の所なぜ，このような現象が起きるのか分かってはいないが，走査集合を求めるのに要する時間はサーバにおける計算時間に比べ小さいことが分かる．**図 5(b)** は，サーバにおける問合せ処理時間を比較したものである．図には，完全なプロトコルとして cPIR を用いた場合の処理時間 (complete) と，検索意図の頻度がガウス分布及び Zipf 分布に従うと仮定して提案手法を用いた場合 (それぞれ gauss, zipf) の三つを記している．多くの場合，サーバにおける計算時間は制約緩和を行った方が完全なプロトコルが要する時間よりも短くて済む．一方，最も高い頻度で問い合わされている検索意図に対しては，完全なプロトコルの方が短い時間であった．これは，ガウス分布における $x$ が $n/2$ 近辺と Zipf 分布における $x$ が 0 近辺のことである．頻度が高い場合，提案手法でもほとんど制約を緩和せずデータベース全体を走査することになる．従って，送られてきた走査集合を解釈するのに要する時間の分だけ，提案手法の方が時間を要することになり，その差が現れているといえる．

## 5. 関連研究

cPIR (Computational Private Information Retrieval) [1] は，計算量をもとに問合せから検索意図が推定できないことを保証するプロトコルである．cPIR における走査集合は，他の多くのプライベート問合せを実現するプロトコルと同様に，検索意図 $x$ によらず常にデータベース全体となる．bbPIR (bounding-box PIR) [3] は，cPIR に $k$-匿名性 [4] の概念を導入することで，サーバにおける問合せ処理時間を改善することを目標として提案された．よって，サーバに課された制約を緩和したプロトコルの一種である．利用者は，問合せ時に検索意図 $x$ を含む幅 $k$ の矩形をサーバに送信する．以降はその矩形に含まれているデータのみからなる仮想的な部分データベースに対して cPIR プロトコルを実行する．従って，問合せ処理時には，データ総数は本来の $n$ から $k$ に削減されており，サーバにおける問合せ処理時間を短縮できる．ただし，1 節にて議論したようにこの制約緩和手法は安全ではない．

暗号化データベース (Encrypted databases; EDBs) [5] も部分的にプライベート問合せを実現している．暗号化データベースでは，すべてのアイテムは暗号化されてデータベースに格納されている．この暗号化されたアイテムの中から検索意図に合致するデータをどのようにして発見するのかが研究の目的である．各アイテムは暗号化されており，問合せも何らかの符号化された値からなる．バケット化 [5], [6] は，アイテムのドメインをいくつかのバケットに分割しアイテムの番号の代わりにそのアイテムが属するバケットの名前を付加する．このアプローチの場合，ある検索意図 $x$ に対する走査集合は，$x$ が属するバケットに等しくなり制約緩和が行われていると見ることもできる．しかし，このバケットの選び方と検索意図頻度の関係，及び問合せのリスクまでを議論している研究は我々の知る限りない．本論文の成果を応用することで安全なバケット分割方法を構築できる．

順保存暗号 (order preserving encryption) を用いた問合せ頻度の秘匿化手法 [7] は，本研究とは異なる方法でプライベート問合せを実現している．そこでは，暗号化後も大小比較が行え，かつ検索意図の頻度とは異なる任意の問合せ頻度を達成できる暗号を用いている．一方で，どの記事が人気であるのかといった検索意図の頻度を攻撃者が知っている場合，暗号化された問合せから検索意図を推定できる確率が高いという問題は議論されていない．また，実際にどのような問合せリスクとなるのかも議論されていな

い．我々の提案手法は，最悪の問合せリスクであっても，完全なプロトコルにおける最悪のリスクを超えないことを保証している点で異なっている．

## 6. おわりに

問合せの頻度情報を基にしたプライベート問合せのための制約緩和手法を提案した．既存のプライベート問合せでは，サーバにデータベース全体を走査させることで安全性を保証しているが，サーバの問合せ処理コストが大きいという問題がある．提案手法は，完全なプロトコルにおけるこの制約を緩和するが，緩和後の問合せリスクは完全なプロトコルにおける最大のリスクは超えないことを保証している．すなわち，我々の緩和手法はサーバにおける問合せ処理コストを削減する一方で，問合せリスクは一般的に許容できると考えられている範囲に収めることができる．

本論文では，ガウス分布と Zipf 分布に従う二種類の問合せを用いて提案手法を評価した．そして，問合せリスクが完全なプロトコルにおける最大のリスクを超えないこと及びサーバによる問合せ処理コストが削減されたことを確認した．さらに，cPIR プロトコルを用いたデータベースを用意し実際に問合せ処理時間を計測することで，提案手法が多くの場合処理時間を短縮することを確認した．

一方，攻撃者が走査集合の大きさを基に検索意図を推定する攻撃が可能かも知れない．なぜなら，現在のアルゴリズムでは問合せ頻度が高いデータを検索する場合，走査集合が大きくなるためである．この問題に対しては，ランダム性を導入し，現在のアルゴリズムによって導き出される走査集合をランダムに大きくするといった対策が考えられるが，今後の課題である．


### 謝辞

本研究は，中島記念国際交流財団及び最先端研究開発プログラム「超巨大データベース時代に向けた最高速データベースエンジンの開発と当該エンジンを核とする戦略的社会サービスの実証・評価」の助成を受けました．


### 参考文献


[1] Kushilevitz, E. and Ostrovsky, R.: Replication Is Not Needed: Single Database, Computationally-Private Information Retrieval, *Proc. of the 38th Annual Symposium on Foundations of Computer Science*, Washington, DC, USA, IEEE Computer Society, pp. 364–373, (1997).

[2] Kawamoto, J. and Yoshikawa, M.: Private Range Query by Perturbation and Matrix Based Encryption, *Proc. of the Sixth IEEE International Conference on Digital Information Management*, Melbourne, Australia, IEEE Computer Society, pp. 211–216, (2011).

[3] Wang, S., Agrawal, D. and Abbadi, A. E.: Generalizing PIR for Practical Private Retrieval of Public Data, *Proc. of the 24th Annual IFIP WG 11.3 Working Conference on Data and Applications Security and Privacy*, Rome, Italy, Springer, pp. 1–16 (2010).

[4] Sweeney, L.: k-anonymity: a model for protecting privacy, *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, Vol. 10, No. 5, pp. 1–14 (2002).

[5] Hacıgümüş, H., Iyer, B., Li, C. and Mehrotra, S.: Executing SQL over Encrypted Data in the Database-Service-Provider Model, *Proc. of the 21th ACM SIGMOD International Conference on Management of Data*, Madison, WI, USA, ACM Press, pp. 216–227, (2002).

[6] Hore, B., Mehrotra, S. and Tsudik, G.: A Privacy-Preserving Index for Range Queries, *Proc. of the 30th International Conference on Very Large Data Bases*, Toronto, ON, Canada, VLDB Endowment, pp. 720–731 (2004).

[7] Agrawal, R., Kiernan, J., Srikant, R. and Xu, Y.: Order Preserving Encryption for Numeric Data, *Proc. of the 23rd ACM SIGMOD International Conference on Management of Data*, New York, NY, USA, ACM Press, pp. 563–574, (2004).


## 付　　録

アルゴリズム 3 の計算量が $O(n^2)$ となることを示す．発見すべき部分行列は四つの変数，左上の位置を表す $(i, j)$，幅，及び高さからなる．本アルゴリズムは，幅 $w$ と高さ $h$ を変えて可能な部分行列の形を列挙する．そして，各形ごとに左上の点を変え，検索意図 $(e, g)$ を含む候補の中で最適な部分行列を探す．

計算量を削減するため，$D'$ の部分行列の和を保持する $C$ と $M$ という二つの行列を用いる．$C$ を列行列と呼び，$h \times 1$ の大きさからなる $D'$ の部分行列の和からなり，$C_{i,j} = \sum_{k=i}^{i+height-1} D'_{k,j}$ である．すべての列の和を保持する代わりに，本アルゴリズムでは，$rangeC[0] \leq i \leq rangeC[1]$, $rangeC[2] \leq j \leq rangeC[3]$ で与えられる $C$ の部分行列のみを保持する．なお，$rangeC$ は今考慮している $C$ の部分行列を (左, 右, 上, 下) の各点からなる四つ組みで記録しており，$rangeC[i]$ は $i$ 番目の値である．$h$ が増加するごとに，各行の和を一要素として $C$ は伸張する．$C$ の更新手続きを updateC と書く．

$M$ は，$h \times w$ の行列で $D'$ の部分行列の和を記録する和行列である．$C$ が $D'$ の要素和から計算されるように，$M$ は $C$ の要素和から計算できる．$C$ と同様に $M$ に対しても考慮すべき $D'$ の部分を $rangeM$ で表す．$M_{i,j}$ は $(i, j)$ 成分を左上とし，高さ $h$，幅 $w$ からなる $D'$ の部分行列内の要素和を保持する．updateM は $M$ の更新手続きである．$C$ と異なり，$w$ が増えると $C$ に記録されている列の和を基に古い $M$ に列を追加することで $M$ を更新するが，$h$ の更新時には $M$ は一から再計算される．

計算結果を再利用できるため，$h$ や $w$ が更新された時に updateC と updateM を計算するコストは $O(n)$ である．$h$ と $w$ の可能な組合せは $n$ 通りであり，updateC と updateM は $O(n)$ 回呼ばれる．従って，アルゴリズム 3 に要するコストは $O(n^2)$ となる．