

# データベースアウトソーシングにおける プライバシー保護に考慮した範囲検索法

新井裕子<sup>†</sup> 渡辺知恵美<sup>††</sup>

<sup>†</sup> お茶の水女子大学理学部情報科学科 〒112-8610 東京都文京区大塚 2-1-1

<sup>††</sup> お茶の水女子大学大学院人間文化創成科学研究科 〒112-8610 東京都文京区大塚 2-1-1

E-mail: <sup>†</sup>yukoarai@db.is.ocha.ac.jp, <sup>††</sup>chiemi@is.ocha.ac.jp

あらまし 近年、データベースの管理運用を外部の技術者に委託するデータベースアウトソーシングの普及に伴い、データ機密保持に対する関心が高まっている。このサービスにおいて管理者は第3者であるため、不十分な管理を行ったり、データを意図的に悪用することも考えられる。そこで、データを暗号化しサービスプロバイダに格納することで、機密を保持する研究が行われてきた。その研究の一つに、2段階暗号を用いた完全一致検索がある [4]。この手法を用いると、管理者に問合せ条件やその結果を知られることなく、サービスプロバイダ側で問合せを実行することができる。本稿では、文献 [4] の手法を拡張し、範囲検索を実現する。本手法では、まず各属性のドメインを複数の領域に分割し、属性値をドメインの最小値からその値までの領域と見なす。範囲検索は、領域同士の包含関係を調べることで実現している。包含の判定にはブルームフィルタを用いている。ブルームフィルタは、キーワード検索を行う際に用いられるビット列である。

キーワード データベースセキュリティ、プライバシー保護、暗号、ブルームフィルタ

## Privacy-Preverving Range Query in Outsourced Databases

Yuko ARAI<sup>†</sup> and Chiemi WATANABE<sup>††</sup>

<sup>†</sup> Department of Information Sciences, Faculty of Science, Ochanomizu University

<sup>††</sup> Department of Natural and Advanced Science, Ochanomizu University

E-mail: <sup>†</sup>yukoarai@db.is.ocha.ac.jp, <sup>††</sup>chiemi@is.ocha.ac.jp

**Abstract** Recently, data confidentiality becomes major concern with the spread of outsourced databases which entrusts database management to an outside administrator. Because an administrator of the outsourced databases is in the third party, users should consider data confidentiality in assumption that an administrator may not have enough skill for managing data or make bad use of the user's data maliciously. To resolve the problem, various mechanisms with cryptographic technologies are proposed. Yang, et al[4] proposed an approach for performing a exact match query without awareness conditions and results of queries to the database administrator by using two-phases encryption method. In this paper, we extend the approach in [4] to perform a range query. In our approach, we first divide each attribute domain into some ranges, and we assume each attribute value as regions from minimal value of the domain to the value. Next we translate the range query condition to the problem of inclusion relation between two ranges. We adopted bloomfilter for the judgment of the inclusion relation. It is a tool of searching keywords quickly.

**Key words** database-security, privacy-preservation, cryptography, bloomfilter

### 1. はじめに

近年、個人情報保護法の制定や大規模な情報漏洩事件が次々に明るみに出たことにより、データベースセキュリティに対する意識が急速に高まっている。多くの DBMS 製品は豊富なセ

キュリティ機能を備えているが、原則としてデータベース管理者に全面的な信頼を置くことを前提に提供されている。そのため、管理不十分による情報漏洩や管理者自身の内部犯行に十分に対処できない。

また、データベース製品の管理運用を外部のデータベース管

理者に委託するデータベースアウトソーシングが現在普及しつつある [2]。このときサービス利用者は、委託業者であるデータベース管理者に機密事項を閲覧されたくないという要求をもつ。

そこで近年では、暗号化したデータと索引をサーバに置くことにより管理者に頼ることなく外部への情報漏洩を防ぐことができ、かつ、管理者に対しても機密を保持できる研究が進められている [1]。また同様のシステムを用いたより安全な索引構成法の研究もある [3]。これは、動的に索引構成を行うことでデータの偏りをなくし、索引解析による内容把握を防ぐものである。別の流れでは、暗号化した文章に対するキーワード検索法 [5] や、それをデータベースに応用した研究がある [4]。これは、元データを 1 回暗号化したものと 2 回暗号化したものをサーバに置き、2 つの差分を用いることで安全に問合せを実行するというものである。

しかし、これらの実用化を考えると、完全一致検索に加え範囲検索や結合演算など様々な問合せに対応できなければならない。そこで本研究では、範囲検索に対応した安全な問合せ法を提案する。この手法は参考文献 [4] の 2 段階暗号とブルームフィルタを用い実現する。

本稿の構成は、第 2 節：データベースアウトソーシングとその攻撃モデル、第 3 節：関連研究、第 4 節：2 段階暗号による暗号化データへの検索、第 5 節：暗号化データに対する範囲検索法とブルームフィルタ、第 6 節：提案手法の改良、第 7 節：実装、第 8 節：性能評価、第 9 節：まとめと今後の課題である。

## 2. データベースアウトソーシングと攻撃モデル

ここでは、データベースアウトソーシングとその攻撃モデルを示す。データベースアウトソーシングとは、データベース管理を専門とする企業がデータ所有者の代わりにデータを管理するサービスのことである。また一般的な攻撃者には、侵入者（システムに対するアクセス権を不法に得て、情報を取りだそうとする者）、内部者（信頼されたユーザグループに属し、そのアクセス権外の情報を得ようとする者）が考えられるが、データベースアウトソーシングの場合、それらに加えてデータベース管理者もまた攻撃者となりうる。管理者が攻撃者となる例として、①怠惰な管理により他のユーザや侵入者に不正な権利を譲渡してしまった場合、②管理者自身が意図的にデータを第三者に売り渡した場合が挙げられる。

管理者または管理者から不正に権利を得たユーザは、以下の攻撃を行う可能性がある。

- 直接的な攻撃：データベースの不正な閲覧・修正・削除
- 間接的な攻撃：ログやシステムカタログ、統計情報の不正な利用
- メモリへの攻撃：サーバに直接アクセスし、メモリ上のデータを不正に利用すること

これらの攻撃は、①データの暗号化（閲覧しても内容把握ができない）、②ログへの配慮（解析からの情報漏洩を防ぐ）、③データ管理の徹底（サーバのメモリ上にヒントを残さない）により対処することができる。

## 3. 関連研究

本節では暗号化されたデータに対する、問合せの手法とキーワード検索に関する研究を紹介する。

SIGMOD2002 にて Hacıgüms 氏は、暗号化されたデータに対する問合せ実行法を提案した [1]。図 1 にクライアントとサーバにおける問合せ処理の流れを示す。データは全てクライアント側で暗号化・索引付与されてから、サーバへ格納される。索引には、その属性のドメイン領域を適当に分割し割り当てられたバケット番号を使う。暗号化されたデータに対する検索は、データに付与された索引を用いて、2 段階問合せにより実現する。これにより、クライアント側での処理が少なくすむため、検索効率をさほど下げることなく問合せを実行することができる。しかし、サーバのデータベーステーブルに付与された索引から大まかな内容が把握できることから、安全性には問題が残る。

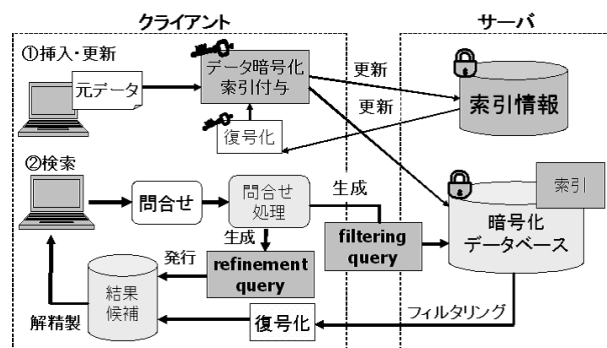


図 1 暗号化データベースに対する処理の流れ

そこで三浦氏らは、安全性に考慮した動的な索引構成法を提案した [3]。この研究では、MaxDiff[6]での領域分割法が用いられている。これは、1 つのバケットに入るタプル数の差が閾値以下になるように分割する方法である。この方法を用いることにより、バケットによるタプル数の偏りが生じなくなるため、分布解析からのデータ内容予測を防ぐことができる。しかしデータ挿入の度に索引情報とデータの更新が行われるため時間コストが高いといえる。

また、別の流れとして暗号化データに対するキーワード検索がある [5]。ここでは、あるキーワードを探索可能にする PEKS という暗号技術が用いられる。図 2 に PEKS を用いた安全なメール振り分けの例を示す。まずキー生成部で公開鍵と秘密鍵を作る。送信者からのメッセージは公開鍵で暗号化される。これを emessage としメールサーバに格納しておく。メール受信者は探索したいキーワードを秘密鍵で暗号化した trapdoor を生成し、これをメールサーバに渡す。サーバはこの 2 つを比較し、受信者が必要とするキーワードを含んだメールのみを返すというものである。この研究をデータベースに応用したものとして、2 段階暗号を用いた問合せ実行法がある [4]。この手法の詳細は 4 章にて示すが、問題点として完全一致検索にしか対応していないことが挙げられる。

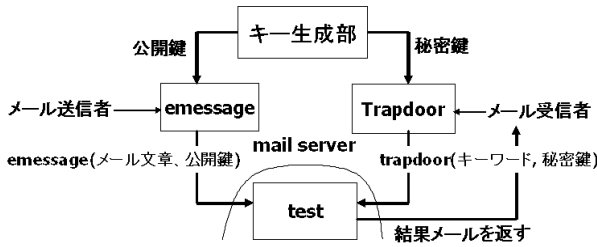


図2 暗号化されたメールに対する振り分け処理

#### 4. 2段階暗号による暗号化データへの検索

本節では、Yang氏らによって提案された2段階暗号[4]について述べる。図3は2段階暗号を用いたデータ挿入と検索の流れを示している。元データの暗号化はセル単位に行われ、その際2つの異なる鍵を用いる。鍵①で暗号化されたデータをデータ1、鍵②で暗号化されたデータをtrapdoor、さらにtrapdoorを鍵としてデータ1を暗号化したものをデータ2とする。これらの処理は全てクライアント側で行うため、管理者でもデータの内容を知ることができない。またデータ1を生成する際、元データに乱数を加えることで、同じ値が同じ文字列へと暗号化されることはない。暗号化処理を終えたデータ1とデータ2はサーバに格納される(図3①)。つまり、1つの属性に対して2つデータがサーバに格納される。暗号化されたデータに対する検索は、クライアントで生成されるcheck1とサーバに格納されているデータ1、データ2を用いて行われる(図3②)。check1とは、ユーザから発行された問合せのwhere文に含まれる値を鍵②で暗号化したものである。例えば、テーブル商品(id, 商品名, 在庫)に対して、ユーザから『select id from 商品 where 在庫 = 30』という問合せが発行されると、check1は30を鍵②で暗号化したものとなる。次にサーバ側でcheck2を生成し、サーバに格納されたデータ2と比較することで、ユーザが発行した問合せの解であるかどうか判別する。check2とはcheck1を鍵としデータ1を暗号化したものである。問合せの解であるとされたタプルはクライアントに返され、復号化の後、ユーザに送られる。

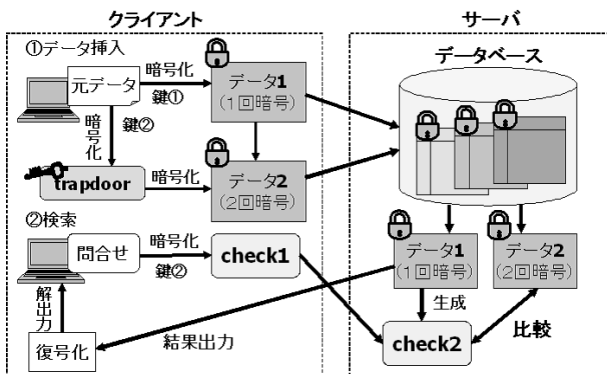


図3 2段階暗号を用いたデータ挿入と検索の流れ

テーブル  $T$  の  $i$  行  $j$  列目のセルを  $T_{ij}$ 、暗号化関数  $E()$ 、乱数  $r_i$ 、2種類の鍵  $k_1, k_2$ 、問合せの where 文に含まれる値を  $x$  とし、各要素を式で示すと次のようになる。

$$\begin{aligned} \text{データ1} &= E_{k_1}(T_{ij} + r_i) \\ \text{trapdoor} &= E_{k_2}(T_{ij}) \\ \text{データ2} &= E_{\text{trapdoor}}(E_{k_1}(T_{ij} + r_i)) \\ \text{check1} &= E_{k_2}(x) \\ \text{check2} &= E_{\text{check1}}(E_{k_1}(T_{ij} + r_i)) \end{aligned}$$

サーバで比較されるデータ2とcheck2の違いは、鍵にある(上式下線部)。trapdoorとcheck1はそれぞれ  $T_{ij}$ 、 $x$  を鍵②で暗号化したものであるから、 $T_{ij} = x(T_{ij}$  が解)であれば、 $\text{trapdoor} = \text{check1}$  となるため、データ2 = check2となり、問合せの解と判定される。一方  $T_{ij} \neq x(T_{ij}$  が解ではない) のとき、 $\text{trapdoor} \neq \text{check1}$  となるため、データ2  $\neq$  check2となり、問合せの解でないとなる。

#### 5. 暗号化データベースに対する範囲検索

本研究では、4節で紹介した2段階暗号をベースとし、そこにブルームフィルタを用いることで範囲検索へと拡張した。また、問合せの一部をサーバで行わせることで、クライアントの負担を減らしている。本節では、まずブルームフィルタの概要を示し、後に提案する範囲検索について述べる。

##### 5.1 ブルームフィルタ

ブルームフィルタとは、ある要素がある集合に含まれるかどうかをテストする際に用いられるビット列のことである。あらかじめ適当な数のハッシュ関数と全て0に設定された空のブルームフィルタを用意しておく。集合の各要素に対するハッシュ値を計算し、空のブルームフィルタに1を立てていく。次に各要素のブルームフィルタの論理和をとり、集合のブルームフィルタとする。テストは集合のブルームフィルタと検索語句のハッシュ値を用いて行われる。検索語句のハッシュ値の全ての位置に1が立っていれば、その要素は集合に含まれていると判断される。

集合  $S = \{\text{東京都, 文京区, 大塚}\}$ 、3種類のハッシュ関数  $hash1, hash2, hash3$ 、8ビットの空のブルームフィルタを例に説明する。図4に集合  $S$  の各要素である『東京都』『文京区』『大塚』から得られたブルームフィルタを示す。集合の各要素のブルームフィルタの論理和をとり、集合  $S$  のブルームフィルタを生成する。次に、このブルームフィルタと検索語句『豊島区』のハッシュ値を比較しテストを行う。検索語句『豊島区』から次のようなハッシュ値が得られたとする。

$$\begin{aligned} \text{hash1(豊島区)} &= 2 \\ \text{hash2(豊島区)} &= 8 \\ \text{hash3(豊島区)} &= 1 \end{aligned}$$

集合  $S$  のブルームフィルタの2,8,1番目のビット列を見ると、2ビット目が0となっているため豊島区は集合  $S$  に含まれないとわかる。

##### 5.2 提案する範囲検索法

図5に提案する範囲検索法の流れを示す。暗号化には4節で紹介した手法を用いる。また属性のドメイン領域を適当に分割

	1	2	3	4	5	6	7	8	
東京都	0	0	1	1	1	0	0	0	論理和
文京区	1	0	0	0	0	1	0	1	
大塚	0	0	1	1	0	0	0	1	
集合S	1	0	1	1	1	0	0	1	

図4 ブルームフィルタの様子

し、それぞれの領域に対してバケット番号を用意しておく。乱数を加えた後、元データを鍵①で暗号化したものをデータ1、バケット番号の各要素を鍵②で暗号化したものを trapdoor、各 trapdoor を鍵としデータ1を暗号化したものをデータ2とする。値によっては trapdoor・データ2は複数となる。次にデータ2からブルームフィルタを生成し、そのブルームフィルタとデータ1をサーバのデータベースに格納しておく(図5:挿入)。検索は、check2とサーバに格納されたブルームフィルタを比較することにより行われる。check1は各バケット番号を鍵②で暗号化したものであり、check2はcheck1を鍵としサーバに格納されたデータ1を暗号化したものである。問合せの解であるとされたタプルはクライアントに返され、復号化・解精製の後、ユーザに送られる(図5:検索)。

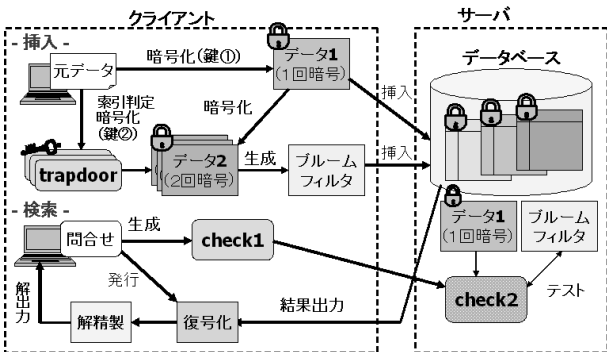


図5 提案する範囲検索法の流れ

### 5.3 挿入例

図6に属性『在庫』の挿入例を示す。元データに適当な乱数  $r_i$  を加え、鍵①で暗号化しデータ1を生成しておく。次にあらかじめ用意しておいた索引情報からバケット番号を割り出す(図6①)。20, 65のバケット番号はそれぞれ  $\{a\}, \{a, b, c\}$  である。各バケット番号を鍵②で暗号化し(図6②), trapdoorを生成する(図6③)。trapdoorを鍵としデータ1を暗号化することでデータ2を生成する(図6④)。データ2からブルームフィルタを生成し、サーバにそのブルームフィルタとデータ1を格納する。

### 5.4 検索例

ユーザから『select id from 商品 where 在庫 >= 30』が発行されたとする(図7)。where文の30のバケット番号  $\{a, b\}$  のbから check1を生成し、問合せを書き換え、サーバに発行する。書き換え後の問合せに含まれる match()関数は範囲検索を行う関数である。詳細は7.3節の検索モジュールにて示す。クライアントから渡された check1を鍵として各行のデータ1を暗

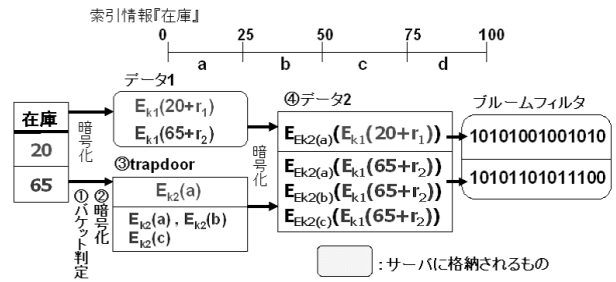


図6 挿入例

号化し、check2を生成する。1つ目のcheck2からハッシュ値6,8,10が得られたとする。1行目のブルームフィルタの6番目には1が立っておらず、解ではないと判断される。一方、2つ目のcheck2からハッシュ値3,5,6が得られたとすると、2行目のブルームフィルタには、いずれにも1が立っているためcheck2は含まれ、この元データは30より大きいと判断される。結果はクライアントに返され、解精製の後ユーザに渡される。

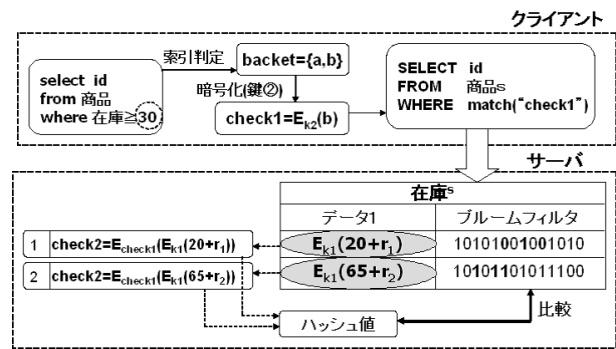


図7 検索例

## 6. 提案手法の改良

前節で提案した範囲検索法には、ドメインの分割数を多くすると検索の精度は上がるが、バケット数が増えるため挿入時間が膨大になるという問題がある。実際、ドメイン  $[0, 10000]$ 、ドメイン領域の分割数2000とし、1000個のレコードを挿入したところ、1レコードあたり5秒ほどかかってしまった。この問題の解決策として我々は多段階の索引構成法を提案する。

### 6.1 索引構成法

1段階分の分割数を  $k$  とした  $n$  段階の索引について述べる。なお分割は等間隔に行うものとする。

図8(a)は  $k=3, n=3$  としたときの索引構成例である。まず、ドメインを  $k$  分割し (level1)、値を含むバケット(図8(a)ではバケットb)をさらに  $k$  分割する (level2)。これを leveln になるまで繰り返す。このような  $n$  段階の索引に対し、該当するバケットとそれより左側にあるバケットからブルームフィルタを生成し、サーバに格納する(図8(a)ではバケットa, b, g, h, m, n, o)。これにより、バケットa及びgに該当する部分のバケット数を節約することができる。改良前の手法で、level3相当の索引を用いた場合、540に対応するバケット数は15個(a:9, g:3, m, n, o)であるが、改良後の索引を用いると7個(a, b, g, h, m, n, o)とな

り、挿入時間の短縮につながる。n 段階の索引、分割数 k の平均バケット数は  $nk/2$  となる。

図 8(b) は、この索引構成例を用いた検索例である。検索条件は 450 以上としている。最大分割数を持つ索引以外 (図 8(b) では level1, 2) では、該当するバケットの右隣りにあるバケット (図 8(b) では c,i)、最大分割数を持つ索引 (図 8(b) では level3) では、該当バケット (図 8(b) では m) から check1 を生成し、検索を行う。

改良前の手法では、検索に用いるバケット数は常に 1 個であったのに対し、改良手法では、n 段階の索引、分割数 k とした場合、検索に用いる最大バケット数は n となり、検索コストは高くなってしまふ。挿入コスト・検索コストは、トレードオフの関係となっているため、適切な分割数、レベル数を検証する必要がある。

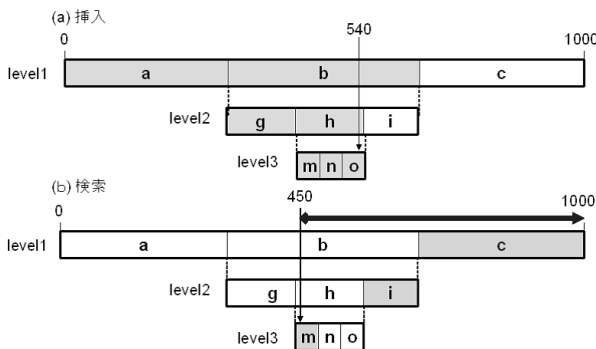


図 8 改良法による挿入と検索

## 7. 実装

範囲検索システムはクライアントで実行される暗号モジュールと reSQL モジュール、サーバで実行される検索モジュールの 3 つから構成される (図 9)。クライアント部分の実装は ruby 言語で行い、RDBMS には PostgreSQL のバージョン 8.2 を用いている。

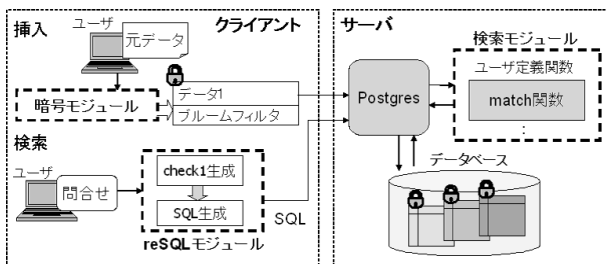


図 9 範囲検索システムの構成

### 7.1 暗号モジュール

暗号モジュールでは、ユーザから提供された元データを 5.3 節で示した手順で暗号化し、データ 1 とブルームフィルタをサーバに格納する。5.3 節同様、属性『在庫』を例に、暗号モジュールの様子を示す (図 10)。暗号モジュールでの具体的な処理は、図 6 に対応している。なお、暗号化関数には openssl の aes256cbc を採用し、ブルームフィルタのサイズは 120 とした。

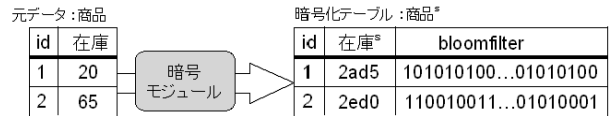


図 10 暗号モジュールの様子

### 7.2 reSQL モジュール

reSQL モジュールでは、ユーザが発行した問合せから check1 を生成し、問合せを書き換え、サーバに発行する。ユーザから以下の問合せが発行された場合を例に、reSQL モジュールの様子を図 11 に示す。reSQL モジュールでの具体的な処理は、図 7 で示したクライアントでの処理に対応している。

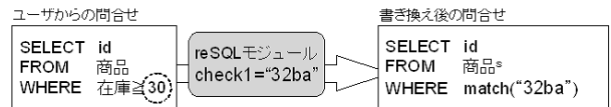


図 11 reSQL モジュールの様子

### 7.3 検索モジュール

検索モジュールでは、reSQL モジュールで書き換えられた問合せを実行し、包含関係を調べることで範囲検索を行う。実際の処理は、問合せに含まれる match() 関数により行われる。match() 関数は、PostgreSQL に定義する新たなユーザ定義関数であり、C 言語で実装した。第一引数には、暗号鍵となる check1 が入り、check2 の生成と包含テストが行われる。前項で示したテーブル 商品\* を例に、検索モジュールの様子とその結果を図 12 に記す。match() 関数の具体的な処理は、図 7 で示したサーバでの処理に対応している。

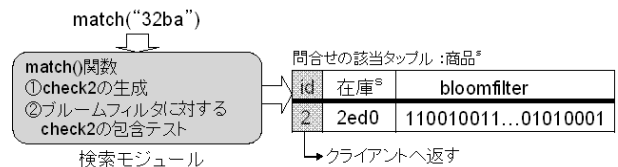


図 12 検索モジュールの様子

①check1 = "32ba" を鍵として、サーバに格納されている属性『在庫』の "2ad5", "2ed0" をそれぞれ暗号化し、check2 を生成する

②check2 のハッシュ値と bloomfilter="110010011...01010001" で包含テストを行い、範囲検索を実行する

## 8. 性能評価

改良前後の手法で性能評価を行った。挿入用データには、[0,1000] の乱数により生成した値を用い、1000 個のレコードを挿入した。また、改良前におけるドメイン領域の分割数を 1000、改良後では 3 段階の索引、ドメイン分割数は 10 とした。7.1 節の暗号モジュールを用いて挿入を行ったところ、改良前の手法では 2887 秒かった。単純に 1 レコードあたりの挿入時間を測定すると、平均 2.887 秒となる。一方、6 節で示した改良法を用いて挿入を行ったところ、結果は 72 秒となり、改

良前に比べて 1/40 と大幅に減少した. 図 13 に改良前後の各問合せに対する実行時間の結果を示す. 1000 個のレコードに対して検索を行った. なお改良後の reSQL モジュールでは, 各段階の該当バケットから check1 を生成し, 各 check1 を引数とした match 関数を or でつなぐことで, 問合せを書き換えている. (Query1,2,3,4: SELECT \* FROM example WHERE value  $\alpha$  ( $\alpha = 800, 600, 400, 200$ ))

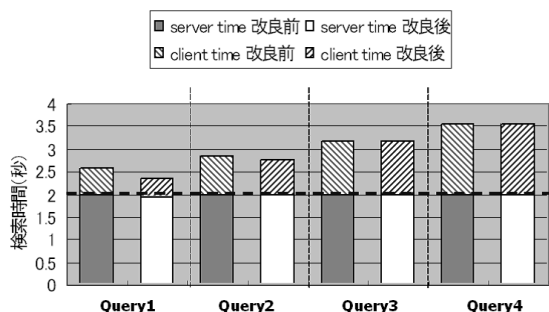


図 13 実行時間

各問合せに対する改良前後のサーバの処理時間は共に 2 秒程度となっている. これは, 問合せ条件である  $\alpha$  が切りの良い値だったため 1 段階目の索引 (level1) のみで検索が完了したためだと考えられる. そこで, 条件を変え検索を行った (図 14). なお各問合せは以下のように設定した. Query1,2,3,4: SELECT \* FROM example WHERE value  $\beta$  ( $\beta = 821, 621, 421, 221$ )

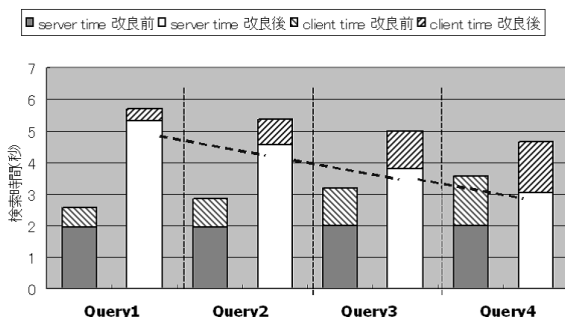


図 14 実行時間

Query1 では, 改良前に比べて 3 倍程度実行時間が増加してしまっている. この理由として, 検索条件が 3 つに増えたこと, 全てのデータに対してシーケンシャルスキャンを行っていることが考えられる. また, 改良後のサーバでの処理時間が徐々に減少している理由については, 221 以上のような解の多い条件の場合, 1 段階目の索引 (level1) で殆どのデータがフィルタリングされるため, 2 段階目 3 段階目の索引を使わなくて済んでいるためだと考えられる. クライアントの処理時間はサーバから返されるタプル数に依存する. 改良前後のサーバから返されるタプル数を図 15 に示した.

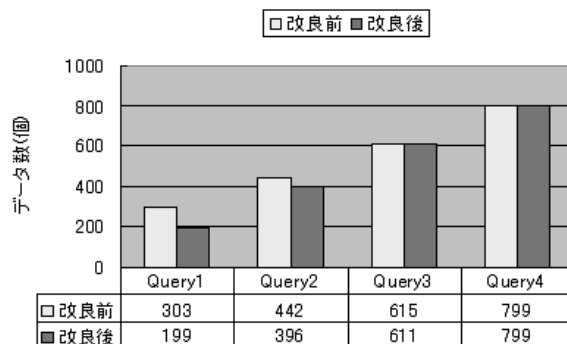


図 15 フィルタリングの結果

## 9. まとめと今後の課題

本稿では, データベースアウトソーシングにおけるプライバシー保護の試みとして, 暗号化されたデータに対する範囲検索法を提案した. データベースの閲覧など強い権利を持つ管理者に対する機密保持に有効であると考えられる. 暗号化の手法には, Yang 氏らにより提案された 2 段階暗号を用いた [4]. また, 値を領域とみなし, ブルームフィルタを用いてその包含関係を調べることで範囲検索を実現した. しかし, データ挿入時の処理数が多く実行時間が膨大となったため, 6 節にて索引構成の改良手法を示した. 改良前後の比較・検証を行ったところ, 挿入時間の大幅な減少が確認できた. 課題としては, 検索時間の高速化, 最適ビット数・バケット数の検証, 以下の条件時に発生する false negative の解消がある. 検索時間の高速化に向けては現在, 1 段階ごとに 2 つのバケットを使い, 無駄な処理を省く手法を考え中である. 今後は, さらに改良を行い, 安全面に関しても提案手法がどの程度のセキュリティに貢献できるかなど検討していきたい.

## 文献

- [1] H. Hacigumus, B. Iyer, C. Li, and S. Mehrotra: "Executing SQL over Encrypted Data in the Database-Service-Provider Model", *Proceeding of the ACM SIGMOD International Conference on Management of Data*, pp. 216-227, June 2002.
- [2] M. Winslett and J. D. Ullman: "Jeffrey D. Ullman speaks out on the future of higher education, startups, database theory, and more," *SIGMOD Record*, 30(3), 2001.
- [3] 三浦志保, 渡辺知恵美: "管理者に対しても機密を保持できる暗号化データベースの索引構成法", 第 18 回データ工学ワークショップ (DEWS2007), E7-8, 2007.
- [4] Zhiqiang Yang, Sheng Zhong, Rebecca N. Wright: "Privacy-Preserving Queries on Encrypted Data", *Proceedings of the 11th European Symposium On Research In Computer Security (Esorics)*, LNCS4189, pp.479-495, 2006.
- [5] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano: "Public key Encryption with keyword Search", *EUROCRYPT*, LNCS3027, pp.506-522, 2004.
- [6] V. Poosala, P. J. Haas, Y. E. Ioannidis and E. J. Shekita: "Improved histograms for selectivity estimation of range predicates," *Proceedings of ACM SIGMOD*, pp.294-305, 1996.