近似的問合せ処理におけるシノプシス構築の高速化

倪　天嘉† 杉浦　健人† 石川　佳治† 陸　可鏡†

† 東海国立大学機構名古屋大学大学院情報学研究科　〒 464–8603 愛知県名古屋市千種区不老町
Email: {ni, lu}@db.is.i.nagoya-u.ac.jp, {sugiura, ishikawa}@i.nagoya-u.ac.jp

あらまし　巨大なデータに基づく問合せを効率的に実行するための技術として，近似的問合せ処理（approximate query processing, AQP）がよく使われている．AQP ではシノプシス（synopsis）を用いた効率的な問合せ処理が行われる．シノプシスとは，対象となるデータをコンパクトに集約したデータを指す一般的な概念であり，さまざまなアプローチがある．近年提案された BAQ（bounded approximate query processing）は元テーブルの適切な部分集合をシノプシスとして保存し，近似誤差がユーザの指定した閾値以下であることを保証する近似的問合せ処理フレームワークである．しかし，BAQ では処理対象となる各数値属性に対してデータをソートする必要があり，数値属性を多数含むようなテーブルのシノプシス構築には時間がかかる．本稿では最小化シノプシスの構築の高速化に焦点を当て既存研究 BAQ を改善し，シノプシスの定義とアルゴリズムを検討する．

キーワード　近似的問合せ処理，問合せ処理，集計計算．

1 はじめに

巨大なデータに基づく問合せを効率的に実行するための技術として，近似的問合せ処理（approximate query processing, AQP）がよく使われている [1–3]．AQP ではシノプシス（synopsis）を用いた効率的な問合せ処理が行われる．シノプシスとは，対象となるデータをコンパクトに集約したデータを指す一般的な概念であり，さまざまなアプローチがある [4–6]．スキャンが一定時間に完了できない大規模なデータベースを対象として，AQP は分析問合せの結果の精度と問合せ処理の待ち時間の間のトレードオフを提供する．

2019 年に Li ら [7] は，一部のデータをシノプシスとして保存する近似的問合せ処理フレームワーク（BAQ）を提案した．BAQ では，ユーザが設定する誤差の閾値と典型的なワークロードを示す問合せ集合を用いてオフライン処理でデータベースからシノプシスを生成し，そのシノプシスを使用して効率的にオンライン問合せに回答する．他の要約データを使用するアプローチ（サンプリング，分位数など [4,8]）と比べて，BAQ は一般的な集計関数（COUNT，SUM，AVG，MIN，MAX）に対して効率的に誤差を保証できる特徴がある．元々提案された BAQ には正の数値データしか扱えないなどの問題点があったため，著者らはそれらの問題を改善した BAQ± を先行研究として提案した [9]．BAQ± では BAQ を元にしてシノプシスの構成要素であるバケット（数値属性の値をグループ化したもの）の生成手法を改善し，正負両方を含む数値属性においても誤差保証を提供すると共に，バケット自体の生成数を減少させることで最終的なシノプシスのサイズも削減した．これにより，元の BAQ に比べより多様な近似問合せに対応するだけでなく，オンラインの近似処理においても実行時間の削減を実現した．

しかし，BAQ± において誤差保証およびオンラインの問合せ処理時間の問題を改善したが，シノプシスの構築時間については改善の余地がある．BAQ± にシノプシスの構築は BAQ と同様に，部分シノプシスの構築と統合シノプシスへの統合の 2 手順に分けられる．部分シノプシスの構築はワークロードの各問合せを対応するため，ワークロードの問合せ数が増えるほど構築時間が増大する．加えて，統合シノプシスへの計算では元テーブルと各部分シノプシスを複数回スキャンする必要があるため，最小化のシノプシスの構築高速化が問題になる．本稿では以上の問題を解決するために，BAQ を元にしてシノプシス構築の高速化に焦点を当て BAQ± の改善アイデアおよび評価実験について述べる．

2 BAQ±

BAQ± は BAQ に基づいて厳密な誤差保証がある高効率な近似的問合せ処理手法を提案した．BAQ± では高精度高効率な近似的問合せ処理を目標として，数値属性のバケット生成手法とオンラインの近似処理手法を改善した．2.3 節では数値属性のバケット分割および 2.4 節ではオンラインでの問合せ処理についてそれぞれ述べる．

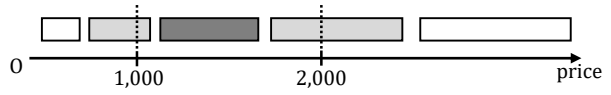
2.1 対象問合せ

BAQ± では以下の演算を用いた OLAP 問合せを想定する．

- COUNT，MIN，MAX，SUM，および AVG での集約
- 一つ以上のカテゴリ属性に対する =，≠ と集約計算
- 一つ以上の数値属性に対する =，≠，>，≧，<，および ≦ を用いた条件での選択と集約計算
- グルーピング

2.2 相対誤差

BAQ± では，BAQ と同様に与えられた問合せに対し，集約結果の真値と近似値との誤差 err として相対誤差を用いる．2 つの値 $x, y \in R$ の相対誤差を以下の式で定める．



条件を部分的に満たすバケット　条件を満たさないバケット
条件を満たすバケット

図 1　バケットの分類（1000 < price < 2000 の場合）

$$err(x,y) = \begin{cases} \frac{|x-y|}{\epsilon} & (x = 0) \\ \left|\frac{x-y}{x}\right| & (\text{otherwise}) \end{cases} \quad (1)$$

ただし，x を真値，y を近似値とし，$\epsilon \in R$ を非常に小さな正の値とする．

2.3　数値属性のバケット分割

BAQ± では，BAQ と同様にユーザがワークロードとなる問合せの集合を用意することが想定されており，与えられた問合せ集合の情報を使用してシノプシスを構築する．各問合せが**問合せの列集約** (QCS) q に変換されて，この QCS を基に各問合せに対応した部分シノプシス S_i が構築される．なお，他の QCS の部分集合となるようなものについては，上位集合となる QCS から生成されたシノプシスによって問合せへの回答が可能となるため，部分シノプシスの生成は行われない．QCS q がカテゴリー属性のみを持つ場合，BAQ± では BAQ と同様にテーブル内のタプルを q に含まれるカテゴリ属性でグループ化することによって構築される．QCS q が数値属性を含む場合，BAQ± ではバケット b について，式 (2) が満たされるようにする．

$$err(\min b, \overline{b}) \leqq \delta \wedge err(\max b, \overline{b}) \leqq \delta \quad (2)$$

まず，QCS q にカテゴリ属性がある場合，テーブル T のタプルをカテゴリ属性でグループ化する．次に，各グループの数値属性の値を昇順にソートする．ソートされた値を式 (2) を満たすバケット $b \subseteq R$ に割り当てる。表 1 で，$q_2 = \{\text{name}, \text{revenue}\}$ および $\delta = 0.05$ を例として説明する．BAQ では{1,300, 1,320, 1,370}を jacket グループにまとめることはできないが，BAQ± では以下のように式 (2) を満たすために単一のバケットとして生成できる．

$$\overline{b} \simeq 1333$$
$$err(\min b, \overline{b}) \simeq err(1300, 1333) \simeq 0.025$$
$$err(\max b, \overline{b}) \simeq err(1370, 1333) \simeq 0.027$$

最終的に表 2 のシノプシスを構築する．

BAQ と比較して，BAQ± ではバケットの対応するデータの範囲はより大きくなる．加えて，BAQ± では 2 つの数値属性に対する選択やグルーピングを含む問合せをサポートする．なお，負の値に対するバケットも同様の方法で生成し，値がゼロのタプルが存在する場合はゼロのみのバケットも追加で生成する．

2.4　問合せ処理

BAQ± では，主にシノプシスを用いて結果を計算する．数値属性の選択がある問合せの場合，数値属性に関する各バケットは**条件を満たす**，**部分的に条件を満たす**および**条件を満たさない**の 3 通りに分けられる．図 1 で，属性 “price” と条件 1000 < price < 2000 の例で説明する．問合せの結果として，条件を満たす場合と部分的に条件を満たす場合の結果を返す．BAQ では部分的に条件を満たすバケット内のタプルが一様分布に従って存在することを仮定しているため，実際には厳密な誤差範囲を保証できない．BAQ に解決されない厳密な誤差範囲を提供するために，提案手法では一様分布の仮定を置かず，条件を部分的に満たすバケットが持ちうる誤差の最大値からシノプシスのみで問合せに回答可能か判定し，誤差を保証できない場合のみ元テーブルを使用する．数値属性 A に対する選択問合せでは，条件を完全に満たすバケットについては，以下の関数で結果を返す．この場合，誤差は発生しない．

- $\texttt{COUNT}(*) \rightarrow \texttt{SUM}(\texttt{SF})$
- $\texttt{SUM}(A) \rightarrow \texttt{SUM}(\text{mean}(A) \cdot \texttt{SF})$
- $\texttt{AVG}(A) \rightarrow \frac{\texttt{SUM}(A)}{\texttt{COUNT}(*)}$

各集約関数について 1 つ以上の数値属性に対する選択・グルーピングを含む問合せについては， [9] の分析により，以上の提案手法は BAQ より小さい誤差となる．

一方で，部分的に条件を満たすバケットについて誤差保証の条件を満たす場合以下の関数で結果を返す． [10] において，各集約関数について部分的に条件を満たすバケットの集約値を計算しやすい近似値で返した際の誤差がしきい値以内となる条件について検討する．

- $\texttt{COUNT}(*) \rightarrow \texttt{SUM}(\texttt{SF})/2$
- $\texttt{SUM}(A) \rightarrow (\texttt{SUM}(\text{mean}(A) \cdot \texttt{SF}))/2$
- $\texttt{AVG}(A) \rightarrow (\frac{\texttt{SUM}(A)}{\texttt{COUNT}(*)})/2$

例として，表 3 のシノプシスについて以下の問合せがある．

表 1　入力テーブル “product”

ID	name	category	revenue
1	jacket	clothes	1,330
2	shirt	clothes	998
3	jacket	clothes	3,120
4	pant	clothes	1,320
5	jacket	clothes	1,370
6	fork	kitchenware	1,000
7	jacket	clothes	1,304
8	shirt	clothes	1,002

表 2　BAQ± で $\pi_2 = \{\text{name}, \text{revenue}\}$ のシノプシス.

name	revenue (min., max., mean)	SF
pant	(1,320, 1,320, 1,320)	1
fork	(1,000, 1,000, 1,000)	1
jacket	(1,300, 1,370, 1,333)	3
jacket	(3,120, 3,120, 3,120)	1
shirt	(998, 1,002, 1,000)	2

表 3　BAQ± で統合シノプシス.

name	category	revenue (min., max., mean)	SF_1	SF_2
jacket	clothes	(1,300, 1,370, 1,333)	5	3
shirt	clothes	(998, 1,002, 1,000)	2	2
jacket	clothes	(3,120, 3,120, 3,120)	0	1
pant	clothes	(1,320, 1,320, 1,320)	0	1
fork	kitchenware	(1,000, 1,000, 1,000)	1	1

```
SELECT name, SUM(revenue)
FROM product
WHERE revenue > 1300
GROUP BY name
```

“shirt” と “fork” グループは条件を満たさない．“pant” グループは条件を満たすバケットのみがあるので，以下の関数で 1,320 の結果を得る．

- $\texttt{SUM}(\text{revenue}) \rightarrow \texttt{SUM}(\text{mean} \cdot \text{SF}_2)$.

“jacket” グループには条件を満たすバケットと部分的に条件を満たすバケットがあるので，BAQ± はシノプシスを使用してそれぞれ計算を行う．条件を満たすバケットは 3,120，部分的に条件を満たすバケットは 3,999 の集計結果がある．結果は誤差保証条件を満たす場合，BAQ± は部分的に条件を満たすバケットについて以下の関数で計算し，合計値 5,120 を “jacket” グループの結果として返す．

- $\texttt{SUM}(\text{revenue}) \rightarrow \texttt{SUM}(\text{mean} \cdot \text{SF}_2)/2$.

3　シノプシスの再定義による構築高速化

本章ではシノプシス構築を高速化するためにまずシノプシスの定義自体を見直し，新たな定義に応じた構築アルゴリズムを提案する．

BAQ± と BAQ におけるシノプシス構築は部分シノプシスの構築と統合シノプシスへの統合の 2 手順に分けられるが，特に部分シノプシスの構築がボトルネックとなる．部分シノプシスの構築は各ワークロード問合せに対して必要となるため，ワークロードとなる問合せが増えるほど処理時間が増大する．特に，全ての QCS がカテゴリ属性しか含まない場合は元テーブルを一度走査するだけで部分シノプシスを構築できるが，数値属性を含む場合はソート処理のために複数回の走査が必要となる．数値属性のソート処理はカテゴリ属性の各グループ，数値属性が複数含まれる場合は各バケットについても行う必要があるため，多数のソート処理が必要となる．

この問題を解決するために，3.1 節では数値属性のバケット化を再定義することでソート処理を排除する手法を提案する．更に，3.2 節と 3.3 節では再定義したバケットに基づくシノプシスの構築手法およびシノプシスの最小化についてそれぞれ述べる．

3.1　バケットの再定義

BAQ においてソート処理が必要となるのは，バケット生成のために各数値属性の最小値が必要となるためである．BAQ においてバケットは相対誤差が閾値 δ 未満となる値の集合であり，言い換えれば最小値と最大値の相対誤差が δ 未満となるような値の集合である．つまり最小値の値が決定した時点でそのバケットの最大値も決まるため，BAQ ではバケット生成の起点としてある数値属性の最小値を求め，そこから昇順に値を処理することでバケットを生成する．

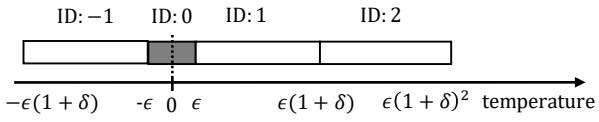


図 2　バケットの再定義

表 4　改善アイデアで q のシノプシス

name	category	revenue.ID	SF
shirt	clothes	142	2
fork	kitchenware	142	1
jacket	clothes	148	2
pant	clothes	148	1
jacket	clothes	149	1
jacket	clothes	165	1

このようなソート処理を除くために，提案手法では各数値属性 A_i に対して最小単位 ϵ_i を導入する．つまり，バケット生成の起点を入力データに応じて動的に決定するのではなく，最小単位を起点とすることでバケット分割を静的に行う．例として，図 2 にこのバケット分割の概略図を示す．まず最小単位として ϵ_i を持つ数値属性 A_i に対し，絶対値が最小単位未満の値（$|a_i| < \epsilon_i$）を ID 0 のバケットとしてまとめる．次に，正の値を含むバケットを以下の式によって ID j にまとめる．

$$[\epsilon_i(1+\delta)^{j-1}, \epsilon_i(1+\delta)^j) \tag{3}$$

同様に負の値を含むバケットは ID $-j$ としてまとめる．

$$(-\epsilon_i(1+\delta)^j, -\epsilon_i(1+\delta)^{j-1}] \tag{4}$$

この分割において最小単位は式 (1) における ϵ に対応し，言い換えればゼロとみなせる閾値を表す．本研究では各最小単位はユーザによって指定されることを想定するが，背景知識に基づき最小単位を決定するのは困難でないと考えられるため，その負荷は問題にならないと考える．

このバケット分割の利点として，ある数値属性の値 $a_i \in A_i$ から対応するバケット ID を以下の式で求められる点が挙げられる．

$$j = \operatorname{sgn} a_i \left\lceil \log_{1+\delta} \frac{|a_i|}{\epsilon_i} \right\rceil \tag{5}$$

なお，sgn は与えた値がゼロのときは 0，正負いずれかの場合は対応する符号（+1 or −1）を返す符号関数である．つまり，元テーブルに含まれる各数値属性の値は静的にバケット化された ID へ変換でき，後述するようにそれに応じたシノプシスの構築が可能となる．

3.2 シノプシスの構築

改善したバケット定義を用いたシノプシス構築について，まずベースラインとして単純な手法を述べる．このベースラインでは BAQ のようなワークロード問合せを用いず，各数値属性の最小単位のみを用いて汎用的なシノプシスを構築する．

ベースライン手法では元テーブルのタプルを順に走査し，各数値属性の値を上述した方法でバケット ID に変換したものをレコードとしてシノプシスへ挿入ないし更新する．このとき，挿入時は対応する SF を 1 に設定し，更新時は 1 ずつ加算することで各レコードの SF を計算する．例えば元テーブルは表 1，誤差の閾値は $\delta = 0.05$，属性 “revenue” の最小単位が 1 であるとき，表 4 のシノプシスが構築される．つまり，ベースライン手法ではバケット ID へ変換された数値属性をカテゴリ属性とみなすことで，全属性を用いた Group By による集約を行っていると言える．

ベースライン手法は単純な手法であるが，構築時間を BAQ よりも大きく削減できる．構築するシノプシスにおいて全属性，つまりカテゴリ属性および変換されたバケット ID の組合せは一意となるため，レコードの挿入ないし更新処理は索引を用いることでレコードの総数 M に対して $O(M)$ で行える．つまり，元テーブルのタプル数が N の場合 $O(N \log M)$ でシノプシスを構築できる．

更に，提案手法はシノプシスの維持管理においても優れた性質を持つ．上記の構築処理は単純な集計処理とみなせるため，分割統治法によって用意に並列化可能である．加えて，BAQ では元テーブルの変更（挿入ないし更新）によって最悪の場合シノプシスの再構築が必要となるのに対し，提案手法は差分更新のみで変更に対応できる．これはバケット ID の存在によって変更の影響が及ぶレコードを一意に特定できるためであり，変更の発生するレコードを更新するのみでシノプシスも元テーブルの変更に追従できる．

3.3 シノプシスの最小化

上述したベースライン手法は単純かつ高速にシノプシスを構築できる一方，構築されるシノプシスのサイズは BAQ よりも大きくなる．ベースライン手法はバケット ID を含む全カテゴリ属性の組合せによってシノプシスのレコードを表現するため，属性数が多い場合，構築されるシノプシスは近似処理に用いるには大きくなりすぎる恐れがある．また，全属性の組合せを考えることでレコードは細分化されるため，実際の問合せ処理において不要な集約が発生することも考えられる．

そこで，BAQ± と同様にワークロード問合せの存在を想定し，シノプシスを最小化する手法について提案する．提案手法は BAQ と同様に各 QCS 集合 q に応じてレコード数 SF_q を設定する．しかし，ワークロード問合せにおいて同時に出現しない属性集合を定義する．提案手法の基本アイデアでは，元テーブルのタプルを順に走査し，シノプシス内で各 QCS に対応する列を順番に埋める．シノプシスに NULL など空きがあるレコードが存在すればタプルの対応値で更新する．シノプシスのレコードに空きがない場合，新規レコードとして挿入する．

表 5　ワークロードに {name, category} および {revenue} 属性集合がある場合

name	category	revenue.ID	SF_1	SF_2
shirt	clothes	142	2	3
fork	kitchenware	148	1	3
jacket	clothes	149	4	1
pant	clothes	165	1	1

具体的には，アルゴリズム 1 により，ワークロード問合せにおいて同時に出現しない属性集合を確認する．バケット ID が計算されたデータを T とする．残り集合と属性を共通しない QCS 集合 q で構成される Q およびデータ T を入力データとし，シノプシス S を初期化する（1 行目）．続いて，順番に T のタプルを各 QCS q によるシノプシス S へ追加する（2–3 行目）．シノプシス S にレコード r の π_q 値がタプル t の π_q 値と同じ場合，対応する SF を 1 増加させる（4–6 行目）．共通しない属性が存在すれば，対応する $\pi_{q'}$ を持つレコードの π_q は NULL の場合 1 つのレコードへ更新する（7–11 行目）．タプル t の π_q 値がシノプシス S に存在しない場合，シノプシス S に π_q 値と $SF = 1$ を追加する（12–14 行目）．

例として，表 1 において，2 つのワークロード問合せに対応する QCS π_1 = {name, category} と π_2 = {revenue} が与えられたとする．QCS π_1 と π_2 は共通する属性を持たないため，それぞれに所属する属性が同時に問合せで用いられることはない．つまり，シノプシス内において各 QCS に対応する列は独立に SF を設定できる．具体的には，π_1 は{ (shit, clothes), (fork, kitchenware), (jacket, clothes), (pant, clothes) }の組合せ，π_2 は{ 142, 148, 149, 165 }についてのみ SF を設定できればよく，それぞれを跨ぐ組合せについては無視できる．この点に基づき最小化することで，シノプシスのレコード数は表 5 まで削減できる．

ただし，異なる π のサイズには大きな差があるので，シノプシスの最小化問題に最適解を求めるのは非現実的になる．本研究では誤差を保証できる近似手法について検討している．

4 実験および評価

本章では，提案手法に関する実装と評価する．BAQ± および BAQ に対し，生成したシノプシスのサイズおよびオフラインのシノプシス生成時間の 2 点で比較する．

4.1 準　　備

a） データセット

本研究では 2 つのデータセットを使用した．1 つ目は 7 列（6 つのカテゴリ列と 1 つの数値列）を持つ Kaggle の temperature データセットである．この 280 万レコードがあるデータセットには，321 個都市の 36 年間の気温の統計が含まれている．2 つ目はよく知られている TPC-H ベンチマークからの scale=1 の 900 万のレコードを含むデータセットであり，4.2 節の実験では 4 列（3 つのカテゴリ列と 1 つの数値列）の orders テーブルを使用する．

Algorithm 1: GenerateSynopsis(T, Q)

Input: T, Q // 元テーブル T と T の QCS
Output: S // 最小化されるシノプシス

```
1  S ← ∅
2  foreach t ∈ T do
3      foreach q ∈ Q do
           // シノプシスに T の QCS を確認
4          if ∃r ∈ S, π_q(r) = π_q(t) then
5              r の SF_q を 1 増やす
6              continue                          // 次の QCS
           // 他のシノプシスに存在しない QCS を確認
7          foreach q' ∈ Q \ q do
               // |q ∩ q'| の降順で QCS を確認
8              if ∃r ∈ S, (π_{q∩q'}(r) = π_{q∩q'}(t)
                 ∧ ∀a ∈ π_{q\q'}(r), a = NULL) then
9                  π_{q\q'}(t) で a ∈ π_{q\q'}(r) を更新
10                 r の SF_q を 1 に設定
11                 break
           // シノプシスに T の QCS がない場合タプルを追加
12         if ∄r ∈ S, π_q(r) = π_q(t) then
13             π_q(t) で r を S に挿入
14             r の SF_q を 1 に設定
```

b） 問 合 せ

実験のため，各データセットについて 310 個の問合せを生成した．これには，選択条件がない MIN/MAX 問合せ 2 個，SUM/AVG 問合せ 2 個，条件がカテゴリ属性のみの COUNT 問合せ 50 個，MIN/MAX 問合せ 50 個，SUM/AVG 問合せ 50 個，選択条件に数値属性を含む COUNT 問合せ 52 個，MIN/MAX 問合せ 52 個，SUM/AVG 問合せ 52 個である．以上の 310 個の問合せ集合によりオフラインにシノプシスを生成して，シノプシスのサイズと生成時間を評価する．加えて，各データについて同時に問合せしない属性集合を設定した．共通しない属性集合は orders データについて{ (status, priority, price), (clerk) }，temperature データについて{(region, city, temperature),(country, year) }で定義した．各数値属性の ϵ を 0.1 とする．

c） 評 価 項 目

提案手法と既存手法を 2 つの角度から比較する：(1) シノプシスのサイズ，(2) オフラインのシノプシスの生成時間（分）．評価実験では，提案手法，BAQ± と BAQ をそれぞれ 30 回実行し，平均値を結果とする．既存手法 BAQ± および BAQ では，部分シノプシスの構築時間および統合シノプシスの生成時間の合計値をオフラインのシノプシスの生成時間とする．加えて，既存研究 BAQ± と BAQ では，スキャンベースの手法および貪欲的な手法で統合シノプシスを生成し，評価する．

4.2 誤差上限に対する性能比較

誤差限界 δ を 0.05，0.1，0.15，0.2，0.25 を設定して，提案手法と BAQ±，BAQ を比較する．

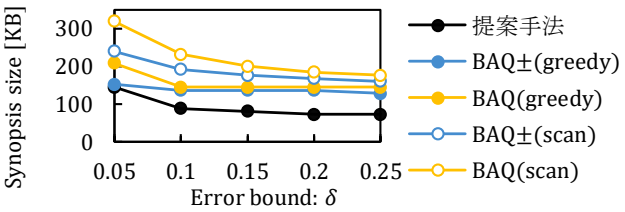


図 3 TPC-H に関するシノプシスのサイズ

4.2.1 シノプシスサイズ

提案手法，BAQ± と BAQ で生成したシノプシスのサイズの実験結果を図 3 と図 4 に示す．

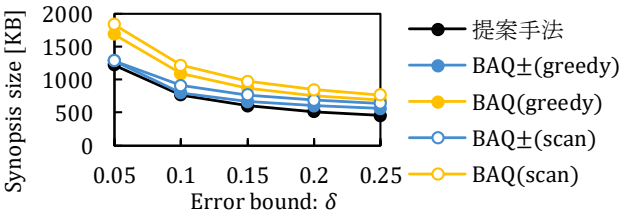


図 4 Temperature に関するシノプシスのサイズ

既存手法が貪欲的な手法でシノプシスを生成する場合，orders データについて提案手法のシノプシスのサイズは BAQ± に対して 56–95%，temperature の場合が 81–95%となっている．BAQ に対して orders データの場合が 50–69%，temperature の場合が 66–72%のサイズでシノプシスを構築できる．既存手法がスキャンベースの手法でシノプシス構築する場合，orders データを使用するとき提案手法のシノプシスのサイズは BAQ± の 45–60%，temperature の場合が 71–95%になった．BAQ より orders データを利用するときは 37–45%，temperature のデータで 59–66%のサイズでシノプシスを生成できる．

以下の分析が行える．提案手法は，実際の問合せ処理において必要な集約のみ考慮してシノプシスを生成する．シノプシスに各 QCS 対応する pi の数に最大数を M_π とし，提案手法は各 QCS に対応する列を順番に埋めるため，シノプシス全体のレコード数は M_π である．しかし BAQ± および BAQ では，複数の部分シノプシスに関して元テーブルのタプルを順に走査し統合シノプシスのレコードとして保存するため，レコード数は M_π より大きい可能性が高い．さらに，誤差の限界を増加させると，各バケット ID の範囲が式 (2) により広くになるため，シノプシスのサイズが減少する．

4.2.2 シノプシス生成時間

オフラインにシノプシスを生成する時間について，orders と temperature データセットに対する計測結果を図 5 と図 6 に示す．

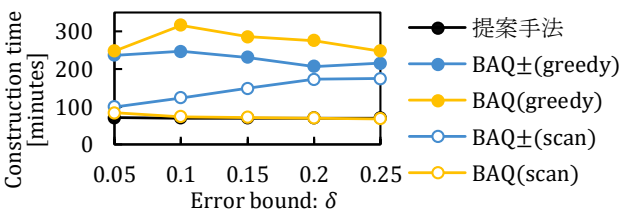


図 5 TPC-H に関するオフラインの生成時間

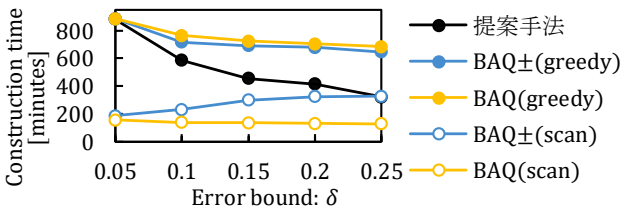


図 6　Temperature に関するオフラインの生成時間

貪欲的な手法を使う BAQ± と BAQ について，orders データを使用するとき提案手法は BAQ± に対して 28–33%，temperature の場合が 49–98%となっている．BAQ より提案手法は orders データの場合が最大 28%，temperature が最大 98%の時間で計算できる．スキャンベースの BAQ± と BAQ について，orders データを利用するとき提案手法は BAQ± の 39–70%になり，BAQ とほぼ同じ結果がある．temperature については BAQ± と BAQ より時間がかかる．

ここで簡単な分析をする．提案アルゴリズムはテーブルの各タプルについて各属性集合 π によるシノプシスをスキャンする．スキャンベースの手法ではテーブルと各シノプシスのスキャンが 1 回だけ必要なため，構築時間が短い．そのために提案手法でシノプシスの構築時間がスキャンベースの BAQ± と BAQ より長い可能性がある．ただし，実験結果により提案手法で計算するシノプシスのサイズは貪欲的な手法とスキャンベースの既存手法より小さいため，最小化シノプシス構築の高速化の目標を達成できると考えている．

5　関 連 研 究

BAQ± [9] と同様に，いくつかの既存の研究は前処理でシノプシスを構築し，そのシノプシスを使用して OLAP 問合せに回答する．ヒストグラム，Wavelets [11] および Sketches [12] 研究は問合せの列集合に基づいてシノプシスを生成し，歪んだデータにも高精度に近似計算できる．ただし，以上の手法では，問合せを全部事前に与える必要がある．なお，これらの研究は数値列のみ適用し，問合せごとにシノプシスが構築されるため，多くのストレージも必要になる．

一方，サンプリングに基づく近似的問合せ処理（SAQ）は AQP に広く研究されている．サンプリングに基づく近似的問合せ処理では，オンラインに問合せの列によるサンプルを抽出し，そのサンプルを使用して結果を推定する．近似結果の精度は抽出されたサンプルに依存するため，結果として良いサンプルを選択するために無作為抽出や層別抽出などの手法が使われている．BlinkDB [13] や VerdictDB [14] 研究では，短くの応答時間を目標としてサンプリングと事前計算を組み合わせて近似的問合せ処理手法を提案する．ただし，BlinkDB と VerdictDB で対応できるのは構造が簡単な問合せのみであり，複雑な述語を含む問合せへの対応は考慮されない．加えて，Ma ら [6, 15] は機械学習モデルを使用してサンプルを抽出し，高精度な近似的問合せ処理システムを開発する．しかし，Ma らの研究では，`MIN` や `MAX` などの一部の集計関数の近似問題を対応できない．サンプリングに基づく近似的問合せ処理では，パフォーマンスと精度の間でトレードオフが実現されるが，いくつかの問題が存在する．まず，サンプリングベースの手法には，BAQ [7] と BAQ± [9] などの厳密な誤差保証がない．加えて，サンプリングベースの手法はロングテールなどの歪んだデータについてパフォーマンスが低下になる可能性がある．つまり，事前に対象データの分布を把握しておくことが必要にある．

6　まとめと今後の課題

本稿では，近似的問合せ処理におけるシノプシス構築の高速化に対する改善アイデアを述べた．また，上記のシノプシスの最小化構築アルゴリズムを実装し，2 つのデータセットを用いて評価実験を行った．実験結果により，提案手法は既存手法 BAQ± よりシノプシスサイズとオフラインのシノプシス構築時間において優れていることを示した．今後の課題としては，今回議論した提案手法に基づく誤差とオンラインの問合せ処理の効能分析とアルゴリズムの改善について詳細な検討を行う予定である．

謝　　辞

本研究の一部は科研費（JP20K19804，JP21H03555，JP22H03594）による．

文　　献

[1] S. Chaudhuri, B. Ding, and S. Kandula, “Approximate query processing: No silver bullet,” in *Proc. SIGMOD*, pp. 511–519, 2017.

[2] K. Li and G. Li, “Approximate query processing: What is new and where to go? A survey on approximate query processing,” *Data Science and Engineering*, vol. 3, pp. 379–397, 2018.

[3] B. Mozafari and N. Niu, “A handbook for building an approximate query engine,” *IEEE Data Eng. Bull.*, vol. 38, no. 3, pp. 3–29, 2015.

[4] B. Walenz, S. Sintos, S. Roy, and J. Yang, “Learning to sample: Counting with complex queries,” in *Proc. VLDB Endowment*, vol. 13, pp. 389–401, 2019.

[5] S. Agarwal, H. Milner, A. Kleiner, A. Talwalkar, M. I. Jordan, S. Madden, B. Mozafari, and I. Stoica, “Knowing When You’re Wrong: Building fast and reliable approximate query processing systems,” in *Proc. SIGMOD*, pp. 481–492, 2014.

[6] Q. Ma and P. Triantafillou, “DBEST: Revisiting approximate query processing engines with machine learning models,” in *Proc. SIGMOD*, pp. 1553–1570, 2019.

[7] K. Li, Y. Zhang, G. Li, W. Tao, and Y. Yan, “Bounded approximate query processing,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 12, pp. 2262–2276, 2019.

[8] B. Ding, S. Huang, S. Chaudhuri, K. Chakrabarti, and C. Wang, “Sample+Seek: Approximating aggregates with distribution precision guarantee,” in *Proc. SIGMOD*, pp. 679–694, 2016.

[9] 倪 天嘉，杉浦 健人，石川 佳治，陸 可鏡, “シノプシスの最適化に基づく近似問合せ処理の高効化,” in **第 *13* 回データ工学と情報マネジメントに関するフォーラム**, 2022.

[10] 倪 天嘉，杉浦 健人，石川 佳治，陸 可鏡, “近似的問合せ処理における問合せ高速化のための誤差保証条件の検討,” in **情報処理学会第 *177* 回データベースシステム研究会**, 2022.

[11] S. Guha and B. Harb, “Wavelet synopsis for data streams: minimizing non-euclidean error,” in *Proc. SIGKDD*, pp. 88–97, 2005.

[12] V. Braverman and R. Ostrovsky, “Generalizing the layering method of indyk and woodruf: recursive sketches for frequency based vectors on streams,” in *Proc. SIGKDD*, pp. 58–70, 2013.

[13] S. Agarwal, A. Panda, B. Mozafari, S. Madden, and I. Stoica, “BlinkDB: Queries with bounded errors and bounded response times

on very large data," in *Proc. EuroSys*, pp. 29–42, 2013.

[14] Y. Park, B. Mozafari, J. Sorenson, and J. Wang, "VerdictDB: Universalizing approximate query processing," in *Proc. SIGMOD*, pp. 1461–1476, 2018.

[15] Q. Ma, A. M. Shanghooshabad, M. Almasi, M. Kurmanji, P. T. Chaudhuri, B. Ding, and S. Kandula, "Learned approximate query processing: Make it light, accurate and fast," in *Proc. CIDR*, pp. 1–11, 2021.