結合を含むクエリに対するシノプシス埋込みよる近似問合せ処理の評価

高田　実佳†　　合田　和生†

† 東京大学情報理工学系研究科　〒 153–8503 東京都目黒区駒場 4-6-1
E-mail: †{mtakata,kgoda}@tkl.iis.u-tokyo.ac.jp

あらまし　業務データを収集し, 蓄積したビッグデータを分析することによる業務の改善・新たな知識発見への期待が高まっている．そのような分析では集計が頻繁に実施される．集計には，必ずしも従来データベースが求める正確な値は必要ではなく，それよりもレスポンスの早さが求められることがある．本論文では，索引構造に付加情報を埋め込むことによって結合を含む問合せに対する近似解を高速検索する方式を既存 RDBMS で実装し，検証する．

キーワード　データベース，索引，近似問合せ

1　はじめに

多くの企業や組織では，日々様々な業務データが生成され，そして収集・蓄積された大量データを分析することによって業務の改善や新たな知識発見に期待が高まっている．業務データの蓄積・管理には構造データベースが広く利用されており，様々な合計値や最大値，最小値など集計値の問合せ処理が分析には必要とされる．例えば，小売業では，日々の売り上げが期待できる製品，製品数や人員を適切に調整する為，製品の購買履歴を用いて，一日の売り上げ総数，在庫数，来客数等の定期的な集計が必要とされている [1]．この様な分析はインタラクティブに実施されることが多く，その為にはデータベースに蓄積されたデータに対し，高速に様々な集計値計算が実行されることが求められる．

高速な集計値計算に向けて，正確な値を高速に求めるデータベースの検索技術はこれまで多数提案されてきた [2]．代表的なものの一つには索引があり，中でも B^+-tree [3] は多くの関係データベースで利用されている． B^+-tree は範囲検索を効率的に行えるメリットがあり集計計算に必要な範囲のデータを高速に検索する為に有効である．しかし，日々業務データが生成・蓄積されることで，そうして増大した業務データを対象とした集計計算の場合，索引のサイズが大きくなり，最下層のリーフページまでノードを辿ると検索時間が増大するという問題が生じている．

インタラクティブな分析における高速な集計値計算を考慮した時に，正確性は必ずしも必要ではない．こうした観点に着目し，[4] らは，既存の B^+-tree など索引にシノプシスを埋込むことで似問合せの高速化手法 (Synopsis-aware search; SAS, Synopsis-aware search plus inter-attribute correlation; SAS+) を提案している．SAS は，B^+-tree の中間ノードに下位ノードの最大値，最小値，合計値，総数といった統計情報をシノプシスとして持たせておく事で B^+-tree のリーフページまで辿らずとも，近似値を求めることができるという近似問い合せ手法であり，SAS+は SAS を発展させ，複数カラム間の相関を考慮してシノプシスの利用可否を決定する近似問合せ手法である．本論文では，[4] の研究を発展させ，業務データの蓄積・管理に広く利用されている関係データベースシステムである PostgreSQL [5] にシノプシスを埋込んだ索引による近似問合せ処理 (SAS, SAS+) を組込み，試作を実装し，単一表に対する問合せに加え，KD-tree を用いて結合を含む問い合わせに対して検証した．

本論文の構成は，以下の通りである．第 2 章では，既存研究とその課題について言及し，第 3 章では，結合を含むクエリに対するシノプシス埋込み索引による近似問合せ処理について説明する．第 4 章では，実装方式について述べ，第 5 章では，評価を示す．第 6 章では，今後に向けた課題と将来展望について纏める．

2　関連文献

オンライン分析処理（OLAP）[6] をはじめ分析クエリ処理は，トランザクションデータが増加する中で分析性能を向上させるために広く研究されてきた分野である．その検索性能を向上させる従来の手法の 1 つとして，データ構造の設計が挙げられる．広く使われるデータ構造としては，検索空間を限定することで高速なデータ取得を可能にする索引が存在する．例えば，B+木 [2,3] のような索引は，指定されたキーを保持し，与えられたクエリ制約に基づいてデータの探索範囲を絞り込むことで効率的にクエリ結果を取得することが可能である．また，また別の手法として，事前計算されたクエリ結果を格納するマテリアライズドビュー [7] が存在する．マテリアライズドビューを利用することで，クエリ処理時には事前計算されたデータを活用し，データ取得・計算コストを削減してクエリ処理を高速化することが可能である．特に，複数の類似したクエリを含むワークロードにおいて，マテリアライズドビューはクエリ性能を向上させる可能性を持つ．しかし，これらのデータ構造はデータ取得性能を効率化する一方で，ストレージのオーバーヘッドやメンテナンスコストを伴うという課題が存在する．そのため，データ取得性能とオーバーヘッドのバランスを最適化すること

が重要である．

一方，厳密な値ではなく近似値を求めることを目的とした近似クエリ処理（Approximate Query Processing, AQP）が研究されてきている．AQP は，主にオンライン処理とオフライン処理の 2 つのカテゴリに分類される．オフライン処理は，保存されたデータからサンプリングされたデータを用いて，クエリ時に集計結果を計算・推定するものである [8–10]．問合せ時にサンプリングデータを用いて近似解を推定し，インタラクティブなクエリの繰り返しを通じて近似解の精度を向上させる技術も提案されている [8]．オンライン処理は，事前処理や事前計算値を保持するためのストレージオーバーヘッドを必要としないという利点を持つが，クエリ時に追加のサンプリング処理が必要となるため，クエリ処理時間が増加する可能性がある．一方，オフライン処理では，統計情報や要約情報などを事前に計算・格納し，クエリ時にその事前計算された情報を利用して近似解を返す方法である [11, 12]．このアプローチでは，シノプシス（要約情報や追加情報）を事前に保持し，それを利用して高速に近似解を返すことが可能である．例えば，ウェーブレットアプローチとして，圧縮された要約情報を保持して検索空間を絞り込む手法が研究されている [13, 14]．また，ヒストグラムをシノプシスとして保持し，データセットの分布を示す方法も存在する [15]．このような事前処理アプローチは，クエリ時の計算コストを削減し，オンライン処理よりも高い近似精度を提供する可能性を持つが，事前計算された追加情報を保持するためのストレージ容量などオーバーヘッドを伴うという課題がある．

近年では，オンライン処理と事前計算アプローチのそれぞれの利点と欠点を考慮し，両者を組み合わせたハイブリッドアプローチが提案されている．BlinkDB [16] は，大規模データにおける他の既存 AQP アプローチと比較して，高精度な近似解を提供することを実証した．BlinkDB は，オンラインのサンプル選択戦略と，ワークロードに基づく多次元層化サンプリングを採用することで，高精度な近似解を取得することが可能である．また，[17] では，シノプシスとサンプリングを組み合わせ，インタラクティブな応答時間で集計値の近似解を返す AQP++が提案されている．さらに，[18] による提案手法では，クエリに応じてオンライン処理またはサンプリングデータを使用するかを判断することで，高精度な近似クエリ処理を実現している．これらのアプローチは高い精度を示しているが，既存のデータベースシステムへの統合が十分に考慮されていないという課題が存在する．

既存データベースシステムへの統合を考慮するため，Hive, Spark，Impala，Redshift などの既存データベースシステムにおいて，より広範なクエリをカバーするクエリリライトを提供するミドルウェアアーキテクチャである Aqua [19]，IDEA [20]，および VerdictDB [21] が提案されている．特に VerdictDB は，既存の SQL ライクなデータベースシステムに対して，システムの変更を加えることなく導入可能であることを実証している．これらのアプローチは本研究の手法と類似しているが，既存のシステムに統合するオーバーヘッドについては十分に議論されていない．

以上を踏まえ，本論文では既存のデータベースシステムにおける近似問合せの適用した際のストレージオーバーヘッドと問合せ実行性能について検証する．

3 結合を含むクエリに対するシノプシス埋込みによる近似問合せ

本章では，シノプシス埋込み索引を用いた近似問合せ処理と，それを単一表のみならず結合を含む問合せに適用する方式について提案する．

これまで，単一表に対するシノプシス埋込み索引を用いた近似問合せ処理については，[4] らは，既存の索引に付加情報であるシノプシスを埋込むことによって近似問合せの高速検索する SAS(Synopsis-aware search) と，属性間の相関を考慮した SAS+(Synopsis-aware search plus inter-attribute correlation) を提案してきた．SAS は，事前にシノプシスを埋め込んだ索引を生成しておき，問合せクエリが入力されると，その索引に埋め込まれたシノプシスを利用する事で探索範囲を狭め，高速に近似解を返す検索方式であり，B^+-tree を用いた評価では，ストレージ負荷が約 2%程度に抑えながら最大 25% 以上近似解の応答時間を高速化を達成できる場合があることを示している．

本研究では，シノプシス埋込み索引を結合を含むクエリに対しても適用する方式を提案する．図 1 を用いて SAS の適用可能性と有効性を説明する．図 1(a) は関係データベースで管理されたテーブルの例 (CUSTOMER 表，ORDERS 表) であり，それらのテーブルの ORDERKEY, CUSTKEY の組み合わせを索引キーとし，付加情報を埋め込んだ KD-tree の例を図 1(b) に示す．図 1(b) には中間ノードとリーフノードを有しており，中間ノードには索引キーとして 1 段目には ORDERKEY，2 段目には CUSTKEY と（水色箇所），下位ノードへのポインタ（灰色箇所），さらに下位ノードの集計値をシノプシスと呼ばれる付加情報として埋め込まれている（橙色箇所）．図 1(b) の例では，付加情報として CUSTKEY 毎の SUM(PRICE) を保有している．リーフノードには，あらかじめ問合せに含まれる結合処理をした結合表のデータあるいはデータへのポインタを保有している．単一表に対する SAS による問合せ処理同様．ルートから深さ毎に対応する索引キーを用いて深さ優先探索を実施し，問合せクエリに対してマッチする集計値を中間ノードで発見した時点で，それより下位のノード検索をスキップする．これにより，データページへのアクセス量を削減し，実行性能が向上できると予想できる．また，オーバーヘッドに関しても，KD-tree は，B^+-tree 同様に，中間ノードは軽量であり，その特性を利用することで，中間ノードに下位ノードの集計値をシノプシスとして埋め込んだとしても，シノプシスによるストレージオーバーヘッドは軽量になると推定できる．以上により，問合せを含む検索においてもオーバーヘッドを抑えて，索引のリーフノードの探索時間を削減し，問合せ処理時間を大幅に短縮できると考える．

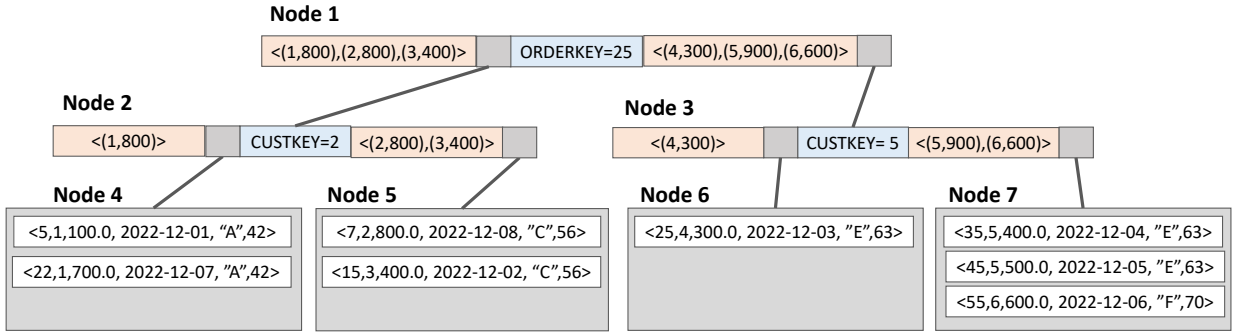
CUSTOMER

CUSTKEY	NAME	AGE
1	“A”	42
2	“B”	32
3	“C”	56
4	“D”	88
5	“E”	63
6	“F”	70
7	“D”	45
8	“E”	36

ORDERS

ORDERKEY	CUSTKEY	PRICE	ORDERDATE
5	1	100.0	2022-12-01
7	2	800.0	2022-12-08
15	3	400.0	2022-12-02
22	1	700.0	2022-12-07
25	4	300.0	2022-12-03
35	5	400.0	2022-12-04
45	5	500.0	2022-12-05
55	6	600.0	2022-12-06

(a) テーブル例



(b) シノプシス埋込み KD-tree の例

図 1: シノプシス埋込み索引の例

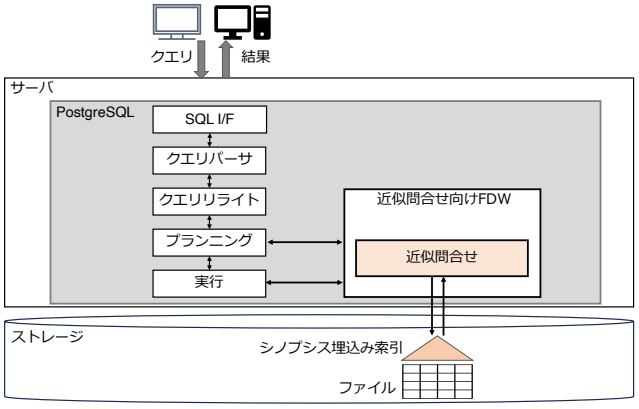


図 2: PostgreSQL 外部データラッパを用いたアーキテクチャ

4 既存関係データベースシステムへの近似問合せ手法の組込み

本章では，既存のデータベースシステムとの併用に向けて，シノプシス埋込み索引を用いた近似問合せ（SAS, SAS+）の実装について述べる．この実装により，従来の索引を用いた検索を超えるクエリ処理性能を実現することを目的としている．

既存のデータベースシステムとして，本研究ではビジネスデータを管理するための最も広く知られたデータベースシステムの 1 つである PostgreSQL [5] を用いた．PostgreSQL は，外部データラッパー（Foreign Data Wrapper, FDW）をサポートしており，外部データを外部テーブルとして定義することで，データベースドメイン外に保存されたデータにアクセスすることを可能にする API を提供している．クエリが与えられると，PostgreSQL エンジンはクエリの解析，書き換え，アクセスプランを生成し，実行して必要な値を取得する．その際に，提供された FDW の API を独自に実装することで，指定の外部データソースにアクセスするためのアクセスプランを生成，および実行プロセスを可能とする．その為，主に 2 種類の API が必要であり，FDW は 1 つはクエリ処理の計画を立てるためのものであり，もう 1 つはクエリ処理を実行するためのものである．本研究では，PostgreSQL は，外部の PostgreSQL にアクセスする為の postgres_fdw [22] を外部データラッパーの 1 つとして公開しており，これを参照し，シノプシス埋込み索引を用いた近似問合せ手法（SAS, SAS+）を実行する，クエリ処理計画用の *GetForeignRelSize, GetForeignPaths, GetForeignPlan* およびクエリ処理実行用の *BeginForeignScan, IterateForeignScan, ReSacnForeignScan, EndForeignScan* を実装した．図 2 は,FDW（Approximate_fdw）を用いて SAI にアクセスする PostgreSQL サーバーのクエリ処理過程を示している，標準 SQL インターフェースを通じてクエリが受け取られると，PostgreSQL エンジンがその解析と書き換えを行う，クエリの対象から，外部テーブルへのアクセスが必要な場合，PostgreSQL サーバーは以下の API を呼び出す．外部テーブルのサイズを推定するための *GetForeignRelSize*，外部テーブルへのアクセスパスを構築するための *GetForeignPaths*，実行プランを作成するための *GetForeignPlan*，そして FDW の下で定義された外部テーブルにアクセスするためのクエリ処理を準備する *BeginForeignScan* ，クエリ処理を実行する *IterateForeignScan, ReSacnForeignScan*，およびクエリ処理を終了する *EndFor-*

eignScan である．本研究では, シノプシス埋込み索引にアクセスするために，シノプシス埋込み索引をストレージ内のファイルシステム上に配置し，それらを用いた近似問合せ処理（SAS, SAS+）を *IterateForeignScan* にて呼び出し，クエリに対して正確な値または近似値を算出し，その値が SQL インターフェースに返される．本研究では，FDW を用いて PostgreSQL にシノプシス埋込み索引を用いた近似問合せ手法を適用し，様々なクエリに対して正確な解または近似解を返す実装アプローチを提案している．設計の詳細は異なる可能性があるものの，類似の設計を他のデータベースシステムにも適用可能であると考えられる．

5 評　　価

本章では，[4] らの提案したシノプシス埋込み索引を用いた近似問合せ手法 (SAS．SAS+) を既存データベースシステムに組込み，その有効性を単一表に対するクエリに対しては B^+-tree, 複数の表に対して結合を含むクエリに対しては KD-tree を用いて検証する．

5.1 実 験 環 境

本実験では，ベンチマークセットである TPC-H [23] の dbgen を用いて生成したデータセットを利用する．データセットのスケールファクタ (SF) は，1,10 を用意した．

次に，問合せ処理に用いる索引として，単一表に対するクエリに対しては，多くのデータベースシステムで適用されている B^+-tree [2, 3] を用い，以下の索引キーをもつ B^+-tree 索引を用意した．全てノードサイズは 8192Byte とした．

- ORDERS (ORDERKEY): ORDERS 表の O_ORDERKEY を索引キーとしてもつ索引
- LINEITEM (ORDERKEY, LINENUMBER): LINEITEM 表の (L_ORDERKEY, L_LINENUMBER) を索引キーとする索引
- LINEITEM (SHIPDATE): LINEITEM 表の L_SHIPDATE を索引キーとする索引
- LINEITEM (PARTKEY): LINEITEM 表の L_PARTKEY を索引キーとする索引.

索引に埋込む付加情報であるシノプシスとしては，以下を用意した．

- **pg-orig**: シノプシス埋込みなし（ネイティブな PostgreSQL）
- **Nosyn**: シノプシス埋込みなし (FDW による実装)
- **Syn1**: 索引キーの集計値 (SUM, MAX, MIN, COUNT) (e.g., ORDERS (ORDERKEY) の場合 O_ORDERKEY の集計値)
- **Syn2**: Syn1 に加え，索引キーとは異なる属性の集計値 (SUM, MAX, MIN, COUNT)
- **Syn3**: Syn2 に加え，索引キーと索引キーとは異なる属性との相関係数および相関係数計算に必要な集計値

本実験では評価クエリとして，単一表の問合せに対してクエリは (Q1-Q4) を用いた．

表 1: 単一表に対するクエリ

Q1	SELECT SUM(L_EXTENDEDPRICE) FROM LINEITEM WHERE L_ORDERKEY BETWEEN 36000000 and 42000000 AND L_SHIPDATE BETWEEN 1992-01-01 and 1992-12-31;
Q2	SELECT MAX(L_EXTENDEDPRICE) FROM LINEITEM WHERE L_ORDERKEY BETWEEN 36000000 and 42000000 AND L_SHIPDATE BETWEEN 1992-01-01 and 1992-12-31;
Q3	SELECT SUM(L_EXTENDEDPRICE) FROM LINEITEM WHERE L_ORDERKEY BETWEEN 36000000 and 42000000 AND L_PARTKEY BETWEEN 1800000 and 2000000;
Q4	SELECT MAX(L_EXTENDEDPRICE) from LINEITEM WHERE L_ORDERKEY BETWEEN 36000000 and 42000000 AND L_PARTKEY BETWEEN 1800000 and 2000000;

さらに，結合を含む問合せに対しては，TPC-H の Q3 を対象として，KD-tree [24] を用いて，索引キーとして (C_MKTSEGMENT, O_ORDERSDATE, L_SHIPDATE) とした．索引に埋込むシノプシスとしては，

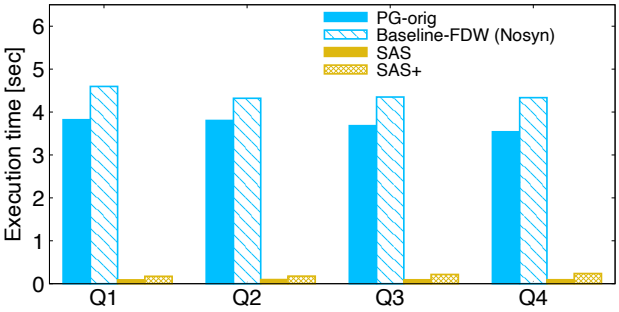
- **Nosyn**: シノプシス埋込みなし
- **Syn**: (L_ORDERKEY, O_ODERDATE, O_SHIPRIORITY) のグループ数と SUM(L_EXTENDEDPRICE*(1-L_DISCOUNT)) を用いた．

本実験では，索引は事前に生成しファイルとして保存し，問合せ時に索引を呼び出し，索引を用いた検索は，IterateForeignScan API 内で実施した．評価クエリでは ORDERS 表および LINEITEM 表を対象としていることから，ORDERS 表および LINEITEM 表を外部表として事前に create する．これにより，外部表に対する問合せクエリを PostgreSQL インターフェース (I/F) から入力すると，近似問合せ処理を実行する FDW(Approximate_FDW) が呼び出される様にした．

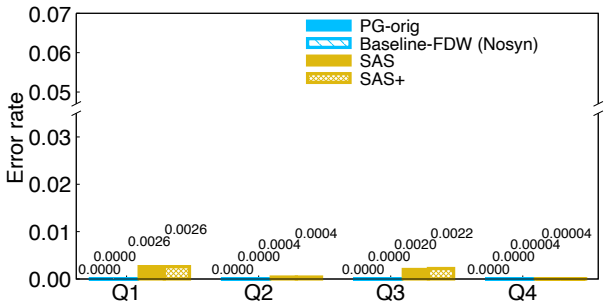
評価軸には，単一表に対する問合せに対して，Native PostgreSQL による B^+-tree 検索 (PG-orig)，FDW を用いた B^+-tree による検索 (Baseline-FDW) と SAS，および相関係数を考慮した SAS+による検索を比較し，エラー率 [18] および問合せに要する実行時間を用いて検証する．また，単一表ならびに結合を含む表に対して各種シノプシスを埋込み索引のリーフノード数，中間ノード数を計測し，シノプシス埋込みによるストレージオーバーヘッドを検証する．なお，全ての実験は，OS は Amazon Linux release 2 (Karoo)，PosgreSQL はバージョン 14.4 を用い，2CPU コア，14GB メモリ，640GB ストレージの環境下で行なった．

5.2 実 験 結 果

初めに，PostgreSQL がもつ既存の B^+-tree を用いた正解値を求める問合せ処理 (PG-orig) と，外部表 (FDW) に対して従来の B^+-tree を用いた正解値を検索する方法 (Baseline-FDW (Nosyn)) と，これらに対して，B^+-tree 索引に **Syn2** のシノプシスを保持し，SAS を FDW(Approximate_FDW) を用いて実行する方式 (SAS) と相関関係を考慮した方式 (SAS+) の実



(a) 実行時間



(b) エラー率

図 3: 単一表に対する近似問合せ処理

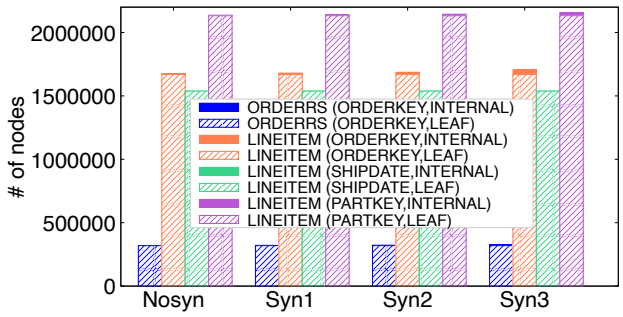


図 4: 単一表に対する索引へのシノプシス埋込みによるストレージオーバーヘッド

行性能を比較検証する．

図 3(a) に単一表に対するクエリ (Q1-Q4) における問合せ実行時間を比較し示す．シノプシスを利用することによって，SAS では約 97.4%，属性間の相関を考慮した SAS+で 94.6% 高速であることが確認できる．また，図 3(b) が示す通り，PostgreSQL に比べ，エラー率は無視できる程度であることが確認できる．本結果は，本実験で用いたデータセットが属性間にほぼ相関を持たず一様に分布している為であると考えられる．

次に，単一表に対する索引へのシノプシスを埋込む事による，索引容量オーバーヘッドを検証する．図 4 に，各索引およびシノプシスをもつデータ構造を保有することによるストレージオーバーヘッドを示す．[4] らの示した結果と同様であり，リーフノードが中間ノードに比べ圧倒的にサイズが大きいため，中間ノードはわずかに増加はしているものの，全体のノード数は，シノプシスによるストレージオーバーヘッドは **Syn3** でも約 1.83% 程度であることを示している．

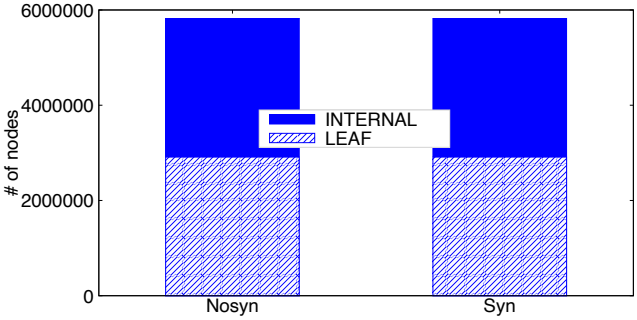


図 5: 結合表に対する索引へのシノプシス埋込みによるストレージオーバーヘッド

さらに，結合を含む問合せに対する結合表に対して索引を生成し，シノプシスを埋込みによる容量オーバーヘッドを検証する．図 5 に示すように，単一表と異なり，複数の検索キーに対応させるため，KD-tree を用いており，リーフページにデータページへのアクセスポインタのみを保有し，問合せに対する結合表をデータページに保有するよう実装したことで，中間ノードとリーフノードのノード数の差分は少ないが，全体のノード数を比較すると，単一表に対する索引同様，シノプシスによるオーバーヘッドは誤差の範囲に留まっている．これより，結合を含む問い合わせに対して，あらかじめ結合表を用意し，それに対しシノプシス索引を埋込んでもオーバーヘッドが少ないことが分かる．

6 おわりに

本論文では，単一表を対象とした問合せに加え，結合を含む問合せに対すして，シノプシスと呼ばれる付加情報を埋込んだ索引を利用し，集合値の問合せに対する近似解を高速に検索する方式を提案し検証した．実験では，単一表を対象とした問合せに対しては，従来の B^+-tree を要する PostgreSQL の検索に比べ，シノプシスを埋め込んだ索引を用いた集合値の計算処理が最大 97.4% 高速である事を確認した．また，単一表に対しても，結合を含む問合せに対してあらかじめ結合表を用意しそれに対してシノプシス埋込んだ場合においても，シノプシスを埋込んだ索引によるストレージオーバーヘッドが無視できる程度であることと，今後は，結合を含む多様な問合せやリアルデータを用いた評価を進める．

謝　辞

本研究の一部は，内閣府戦略的イノベーション創造プログラム（SIP）「統合型ヘルスケアシステムの構築」の助成を受けたものである．

文　献

[1] 川村晃一．売上分析．計測と制御, Vol. 28, No. 1, pp. 17–22, 1989.
[2] Ramez Elmasri. Fundamentals of database systems seventh edition. 2021.
[3] David J Abel. A b+-tree structure for large quadtrees.

Computer Vision, Graphics, and Image Processing, Vol. 27, No. 1, pp. 19–31, July 1984.

[4] Hiroki Yuasa, Kazuo Goda, and Masaru Kitsuregawa. Exploiting embedded synopsis for exact and approximate query processing. In *Proc. DEXA*, pp. 235–240, 2022.

[5] Postgressql. `https://www.postgresql.org/`. 2024-01-10 参照.

[6] Surajit Chaudhuri and Umeshwar Dayal. An overview of data warehousing and OLAP technology. *SIGMOD Rec.*, Vol. 26, No. 1, pp. 65–74, 1997.

[7] Imene Mami and Zohra Bellahsene. A survey of view selection methods. *SIGMOD Rec.*, Vol. 41, No. 1, pp. 20–29, 2012.

[8] Joseph M Hellerstein, Ron Avnur, Andy Chou, Christian Hidber, Chris Olston, Vijayshankar Raman, Tali Roth, and Peter J Haas. Interactive data analysis: the control project. *Computer*, Vol. 32, No. 8, pp. 51–59, 1999.

[9] Chris Jermaine, Subramanian Arumugam, Abhijit Pol, and Alin Dobra. Scalable approximate query processing with the dbo engine. *TODS*, Vol. 33, No. 4, pp. 1–54, 2008.

[10] Kai Zeng, Sameer Agarwal, Ankur Dave, Michael Armbrust, and Ion Stoica. G-ola: Generalized on-line aggregation for interactive analysis on big data. In *Proc. SIGMOD*, pp. 913–918, 2015.

[11] Niranjan Kamat, Prasanth Jayachandran, Karthik Tunga, and Arnab Nandi. Distributed and interactive cube exploration. In *Proc. ICDE*, pp. 472–483, 2014.

[12] Swarup Acharya, Phillip B Gibbons, Viswanath Poosala, and Sridhar Ramaswamy. The aqua approximate query answering system. In *Proc. SIGMOD*, pp. 574–576, 1999.

[13] Ioannis Mytilinis, Dimitrios Tsoumakos, and Nectarios Koziris. Distributed wavelet thresholding for maximum error metrics. In *Proc. SIGMOD*, pp. 663–677. Association for Computing Machinery, 2016.

[14] Abdul Naser Sazish and Abbes Amira. An efficient architecture for HWT using sparse matrix factorisation and DA principles. In *Proc. APCCAS*, pp. 1308–1311, 2008.

[15] Graham Cormode, Antonios Deligiannakis, Minos Garofalakis, and Andrew McGregor. Probabilistic histograms for probabilistic data. *Proc. VLDB Endow.*, Vol. 2, No. 1, pp. 526–537, August 2009.

[16] Sameer Agarwal, Barzan Mozafari, Aurojit Panda, Henry Milner, Samuel Madden, and Ion Stoica. BlinkDB: queries with bounded errors and bounded response times on very large data. In *Proc. EuroSys*, pp. 29–42, 2013.

[17] Jinglin Peng, Dongxiang Zhang, Jiannan Wang, and Jian Pei. AQP++: Connecting approximate query processing with aggregate precomputation for interactive analytics. In *Proc. SIGMOD*, pp. 1477–1492, 2018.

[18] Xi Liang, Stavros Sintos, Zechao Shang, and Sanjay Krishnan. Combining aggregation and sampling (nearly) optimally for approximate query processing. In *Proc. SIGMOD*, pp. 1129–1141, 2021.

[19] Swarup Acharya, Phillip B Gibbons, Viswanath Poosala, and Sridhar Ramaswamy. Join synopses for approximate query answering. In *Proc. SIGMOD*, pp. 275–286, 1999.

[20] Alex Galakatos, Andrew Crotty, Emanuel Zgraggen, Carsten Binnig, and Tim Kraska. Revisiting reuse for approximate query processing. *Proc. VLDB Endow.*, Vol. 10, No. 10, pp. 1142–1153, 2017.

[21] Yongjoo Park, Barzan Mozafari, Joseph Sorenson, and Junhao Wang. Verdictdb: Universalizing approximate query processing. In *Proc. SIGMOD*, pp. 1461–1476, 2018.

[22] postgres_fdw. `https://www.postgresql.jp/document/14/html/postgres-fdw.html`. Accessed on 02-26-2024.

[23] Tpc-h. `https://www.tpc.org/tpch/`. 2024-01-09 参照.

[24] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, Vol. 18, No. 9, pp. 509–517, 1975.