シノプシス埋込みによる近似問合せ処理の試作実装と初期評価

高田　実佳† 　喜連川　優† 　合田　和生†

† 東京大学　〒 153–8503 東京都目黒区駒場 4-6-1
E-mail: †{mtakata,kitsure,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問合せに対する近似値を高速に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．このような分析では，多くの場合，合計値や最大値，最小値などの集計値が求められる．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]．このような集計値の計算には，正確な値は必ずしも必要ではなく，インタラクティブに多角的な分析を実施するため，レスポンスの早さがより求められることがある．

高速な集計値計算に向けて，正確な値を高速に求める為の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，そのデータ量が増えるにつれて索引のサイズも大きくなる為，最下層のリーフページまでノードを辿る場合，検索時間が増大するという問題が生じる．

一方，正確な値ではなく近似値を求める方法として近似問合せ処理 (AQP; Approximate Query Processing) が研究されてきた [4,5]．既存の近似問合せ手法は大きく分けて 2 通りに分類されており，一つはデータベース中のサンプルデータのみを用いてオンラインに集計計算を実施する方式，もう一つは事前にデータベースのデータからシノプシスという付加情報を算出しておき，問合せ時にその付加情報を用いて近似値を返すという方式である．しかし，前者はストレージ負荷はかからないが問合せのタイミングでサンプルデータの抽出が計算処理に加算されるというデメリットがあり，後者は前者に比べ事前計算した付加情報を用いることで高速だが，付加情報を保存しておくというストレージオーバーヘッドを要するというデメリットがある．[6] らは，既存の B+Tree 索引にシノプシスを埋め込むことで似問合せの高速化手法 (SAS; Synopsis-aware search) 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提案である．

本論文では，[6] の研究を発展させ，既存データベース [7] のインターフェースを利用して集計値を問合せるクエリを入力し，近似問合せの結果を得る試作を実装し，初期評価結果を示す．

本論文の構成は，以下の通りである．第 2 章では，既存の近似問合せとその課題について言及する．第 3 章では，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し，第 4 章では，その初期評価を示す．第 5 章では，関連文献について言及し，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．

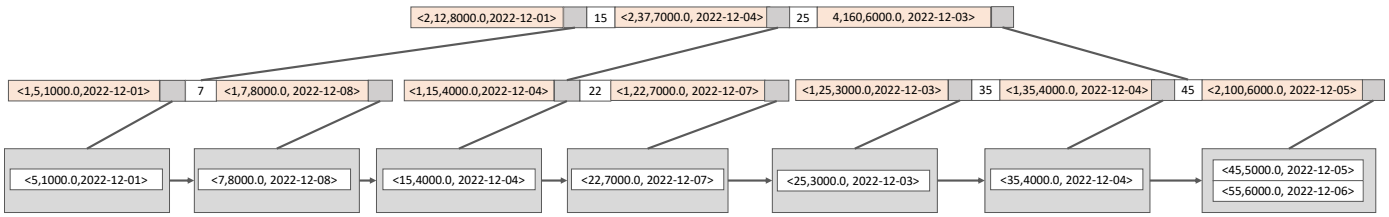
2　近似問合せ処理

本章では，近似問合せ処理へのニーズと，既存の近似問合せ処理およびその問題点について述べる．近年，ビックデータを用いた分析に期待が高まり，広くインタラクティブな集計計算が実施されている．例えば製造業では，IoT センサ等の発達により，原材料の調達，設計，製造，流通，といった製品のライフサイクルの管理のデータを計測でき収集・蓄積が進んでいる [8]．蓄積された大量のデータを利用し，日々の受注製品数，最大受注製品，売上金額なと集計値を定期的に計算し，その値に基づき，調達すべき原材料の種類，個数や，製造すべき生産量，出荷量，販売量の適切な決定に活用している [9]．このようにデータを活用した業務改善・効率化には，統計的な集計値の計算が頻繁に実施される必要がある．この様な集計計算には，必ずしも一円単位の売り上げ金額や一桁単位の生産数といった正確な値は必要ではなく，それ以上に様々な種類の集計値をインタラクティブに実行できることに価値が求められる場合がある．即ち，合計値，最大・最小値，平均値など様々な集計値の問合せに対してインタラクティブなスピードで結果を返すことが求められる．

集計値の近似値を求める近似問合せ処理はこれまで長年議論されてきた．代表的な手法は 2 通りに分類され，蓄積された

ORDERKEY	PRICE	SHIPDATE
5	1000.0	2022-12-01
7	8000.0	2022-12-08
15	2000.0	2022-12-02
22	7000.0	2022-12-07
25	3000.0	2022-12-03
35	4000.0	2022-12-04
45	5000.0	2022-12-05
55	6000.0	2022-12-06

(a) テーブル例



(b) シノプシス埋込み B+Tree の例

図 1: シノプシス埋込み索引の例

データからサンプリングしたデータのみを用いて問合せ時に即座に集計値を計算・推定する方式であり，もう一つは事前にデータベースに蓄えられたデータを用いて集計値など付加情報を計算し保存しておくことで，問合せ時にその値を用いて集計値を返答する方式である．しかし，前者は事前準備が不要のため事前準備に時間や付加情報を保村しておくためにストレージ負荷がかからないが応答時間が長いというデメリットがあり，後者は前者に比べ応答時間が高速だが，付加情報保存のためのストレージ負荷がかかるというデメリットがある．この様な問題に対し，[6] は既存の B+Tree 索引にシノプシスを埋め込むことによる近似問合せの高速化手法 (Synopsis aware search; SAS) を提案している．SAS は，事前に付加情報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，その索引を用いて検索し，付加情報を利用する事で高速に近似解を返す検索方式であり，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．

図 1 を用いて SAS の有効性を説明する．図 1(a) は関係データベースで管理されたテーブルの例であり，そのテーブルの ORDERKEY を索引キーとし，付加情報を埋め込んだ B+Tree の例を図 1(b) に示す．図 1(b) では，一般的な B+Tree に加え，その各中間ノードには，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．図 1(b) の例では，COUNT(ORDERKEY)， SUM(ORDERKEY)，MAX(PRICE)，MIN(SHIPDATE) を保有している．一般的に B+Tree はリーフにレコードを持つためデータサイズが大きくなるが中間ノードは軽量である．その特性を利用し，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でいるが，オーバーヘッドは軽量になると見込まれる．そして SAS による検索処理では．一般的な B+Tree を用いた探索同様，ルートから深さ探索を実施し，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停止できる．これにより，問合せ検索において索引のノード探索時間を大幅に短縮できることが検証されている [6]．しかし，SAS は事前に準備した付加情報が利用できる場合においては非常に有効であるが，ユーザが求める多様な集計値の問合せに対してシノプシス埋込み索引を準備する時間やストレージ等のオーバーヘッドについては．検討が十分行われていない．この問題に対し，本論文では，ユーザの求める多様な集計値の問合せへの対応を可能にする高速な近似問合せ処理を提案する．

3 既存関係データベースへの近似問合せ手法の組み込み

本章では，ユーザの多様な集計値計算ニーズに応える為，[6] らの既存索引の中間ノードにシノプシスを埋め込むことによる高速な近似問合せ処理方式を既存データベースに取り込む事によって，近似解を高速に応答する方式を提案する．

まず初めに，ユーザの多様な集計値計算ニーズに応える為，広く利用されているオープンソースの関係データ管理システムである PostgreSQL [7] へのシノプシスを埋め込んだ索引 (Synopsis-alloyed tree) を取り入れる実装の実現性を検討する．通常，データベース管理システムは，データベース構築時に指定したデータ領域に存在しているデータにアクセスすることが一般的である．PostgreSQL はそれに加え，外部データラッパー (FDW; Foreign Data Wrapper) をサポートしており，その機能により外部サーバ (Foreign Server) やその外部サーバに存在している外部テーブル (Foreign Table) を登録しておくことで，PostgreSQL インターフェースを用いて外部サーバにある外部テーブルを参照できる機能である [10]．様々な種類の外部データラッパーは既に公開されており，例えば，外部のファイルにアクセスして検索する File FDW や別の PostgreSQL サーバにアクセスする Postgre FDW などが既に公開されている．その

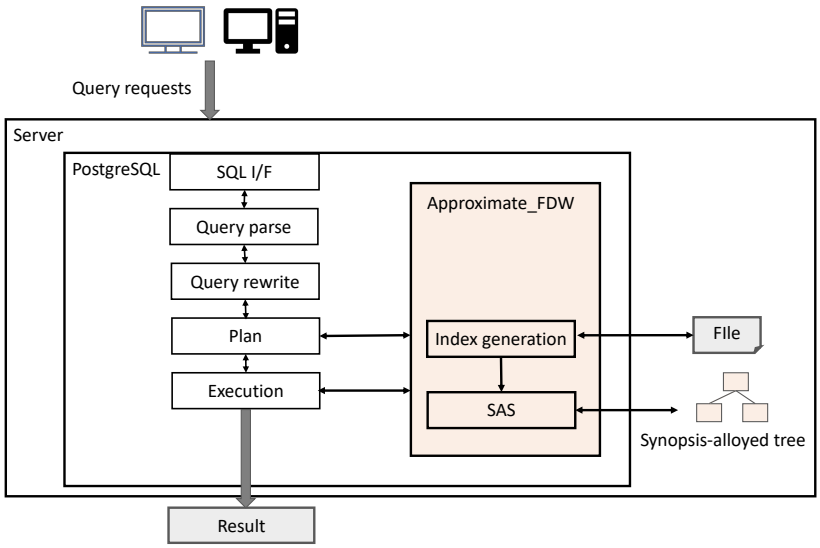


図 2: PostgreSQL 外部データラッパを用いたアーキテクチャ

様な公開済み FDW に加え，PostgreSQL は独自の FDW を開発できるよう API を提供しており，我々はその API を利用する事で近似問合せ処理を実施する独自 FDW を開発する方式を検討に取り入れた．以下その独自 FDW を Approximate FDW とよぶ．

PostgreSQL は FDW 作成に必要な API を提供しており，我々は今回その中から下記の 7 種類の API を用いる事とする．各 API の概要は以下の通りである．

- GetForeignRelSize: テーブルサイズを見積もる
- GetForeignPaths: アクセスパスを生成する
- GetForeignPlan: プランノードを生成する
- BeginForeignScan: スキャン開始時に呼ばれる
- IterateForeignScan: 上位ノードが 1 行必要とした時に呼ばれる
- ReSacnForeignScan: スキャン位置を先頭に戻したい時に呼ばれる
- EndForeignScan: スキャン終了時に呼ばれる

これら API は PostgreSQL に実行計画と実行処理の 2 種類に大別できる．GetForeignRelSize, GetForeignPaths, GetForeignPlan が実行計画に関連する API であり，BeginForeignScan, IterateForeignScan, ReSacnForeignScan, EndForeignScan が実行処理に関連する API である．ユーザは問合せ前に，PostgreSQL サーバに Approximate FDW，Approximate FDW を用いる外部サーバ，外部テーブルの登録を行うことで，外部テーブルへの問合せを PostgreSQL が認識した際，Approximate FDW を呼び出す事ができる．下記に例を示す様に，外部テーブルは通常テーブル定義と同様に SQL を用いて登録が可能である．

- Approximate FDW(e.g., approximate_fdw) 登録

```
CREATE EXTENSION approximate_fdw;
```

- Approximate FDW を用いる外部サーバ (e.g., approximate_server) 登録

```
CREATE SERVER approximate_server
    FOREIGN DATA WRAPPER approximate_fdw;
```

- 外部テーブル (e.g., foreign_table) 登録

```
CREATE FOREIGN TABLE foreign_table (val int)
    SERVER approximate_server;
```

この様な FDW 機能を用いて，シノプシスを埋め込んだ索引を用いた検索を実現するアーキテクチャを図 2 に提案する．ユーザが PostgreSQL のインターフェースを介してクエリを投入すると，図 2 に示すように，クエリはパース (Query parse)，クエリの書き換え (Query rewrite) を経て，実行計画 (Plan)，実行 (Execution) と処理される．そして，事前に登録した外部テーブルへの問合せが検知されると，PostgreSQL は対応する FDW を呼び出し，FDW 作成時に登録した実行計画，実行処理を行うので，実行処理内で，シノプシスを埋め込んだ索引を生成し，その索引を利用した検索が可能である．これによりユーザが与えた問合せクエリに対し，独自の処理方式を用いて求めたレコードを返す事ができる．以上の仕組みを用いて，我々は問合せ時にシノプシス埋込み索引を構築し，その索引を用いて集計計算に対する近似値を求めるプロトタイプを開発した．

4 初 期 評 価

本章では，[6] らの近似問合せ処理である SAS が既存データベース上で実行可能か，そしてその優位性を検証する為，第 3 章で述べた様に PostgreSQL の FDW を利用してシノプシスを埋め込んだ索引および SAS を実装し，その初期評価結果を示す．

4.1 実 験 環 境

まず初めに実験に用いたデータセットについて述べる．スケールファクタ 1(約 1GB) の TPC-H [11] データセットを dbgen を用いて生成し，ORDERS 表を利用する．索引には，B+Tree [2,3] を用いて，O_ORDERKEY を索引キーとし，ノードサイズは

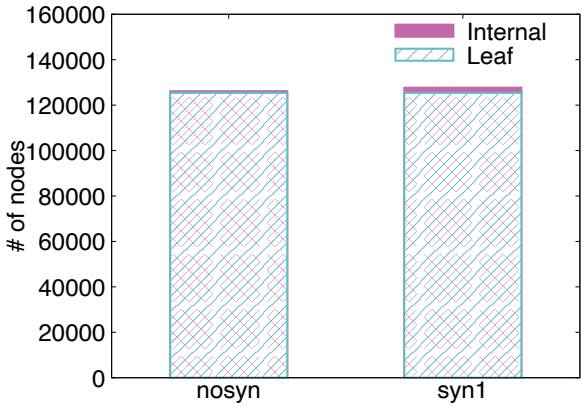


図 3: シノプシス埋込みによるストレージオーバーヘッド

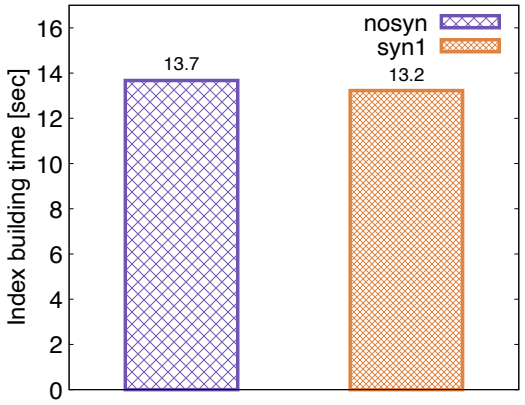


図 4: 索引準備時間オーバーヘッド

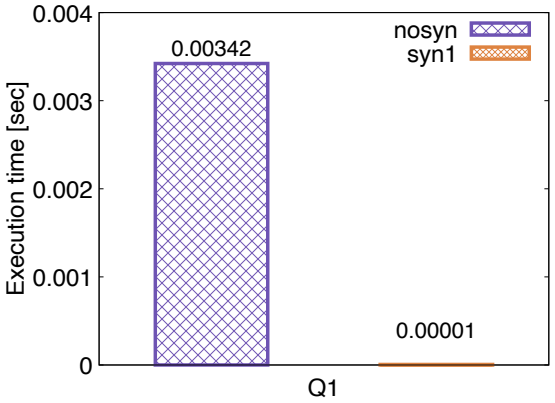


図 5: 近似問合せ実行時間

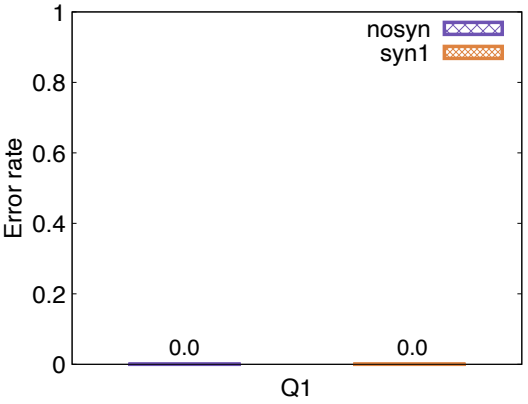


図 6: 近似問合せ結果の相対エラー率

4096B とする．付加情報であるシノプシスは，

- nosyn: シノプシス埋込みなし
- syn1: 索引キー (O_ORDERKEY) の集計値 (SUM, MAX, MIN, COUNT)

2 種類を用意する．本実験では，索引の生成は，3 章で述べた BeginForeignScan API 内で実施し，生成した索引を用いた問合せに対する検索は，IterateForeignScan API 内で実施する様に実装した．これにより，Approximate FDW が呼ばれた際，BeginForeignScan API が実行処理の最初に呼ばれ，索引の生成が実施される．

集合値の問合せクエリには，

Q1: SELECT COUNT(O_ORDERKEY) FROM ORDERS WHERE O_ORDERKEY BETWEEN 5400001 AND 6000000;

を用いる．本クエリは ORDERS 表を対象としている為，本実験では事前準備として，今回開発した FDW である Appproximate FDW，Approximate Server とそれに属する外部テーブルとして ORDERS 表を事前に create 文を用いて登録しておく．これにより，ORDERS 表に対する問合せクエリを PostgreSQL インターフェースから入力すると，Approximate FDW が呼び出される様にした．

評価軸には，各 nosyn，syn1 のシノプシスを埋め込んだ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，索引生成に要するストレージオーバーヘッドと索引準備時間オーバーヘッドを検証する．また，集計値の問合せに対して，一般的な B+Tree の検索と SAS による検索を比較し，相対エラー率 [12] および集計計算に要する実行時間を用いて検証する．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，32GB ストレージの環境下で行なった．

4.2 実験結果

初めに，シノプシス埋込みによる，索引生成に要するオーバーヘッドを検証する．図 3 に，ストレージオーバーヘッド，図 4 に索引準備時間オーバーヘッドを示す．図 3 に，ストレージオーバーヘッドは，[6] 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およそ 1.3% 程度であることを示している．また，図 4 からシノプシスを埋め込むことによる索引生成に要するオーバーヘッドはほぼ無と言えることが分かる．

次に，PostgreSQL 上での SAS による近似問合せの有効性を検証する．図 5 に，集計計算に要する実行時間結果，図 6 に相対エラー率を示す．これは，実験に用いた Q1 では syn1 のシノプシスを十分利用できる為，シノプシス埋込みがない索引 (nosyn) による検索に比べ，リーフノード探索時間を大幅に削減する事ができ，500 倍以上高速でありながら，近似解の誤差

も生じていない事を示している．

5 関 連 文 献

大量データを収集・蓄積できるようになるに従い，蓄積した大量データから集計値を高速に計算する手法は長く研究されてきた．分析のための高速検索手法は OLAP [13] 問合せとして提案されてきた．代表的なものに索引があり，B+Tree [2,3] をはじめ検索キーを検索に効率的なデータ構造で保持することで高速検索を実現する．Data cube [14] は Group-by など集計に必要な処理をオペレータとして用意することによる処理の効率化を提案している．Materialized View [15] は過去の問合せ結果やその中間データを保存しておくという手法であり，類似する問合せを多く実行する分析系の問合せにおいて大幅な高速化が可能である．Materialized View は大規模データを対象とした検索時間を大幅に削減できるが，その効果の反面，大規模な Materialized View を保存するとストレージコストやメンテナンスコストが増加してしてしまうため，高速性とのバランス調整が求められる．

一方，正確な値ではなく近似値を求める方法として近似問合せ方式があり，それは 2 通りに大別される．1 つは蓄積されたデータからサンプリングしたデータのみを用いて問合せ時に即座に集計値を計算・推定するオンライン処理型方式であり，もう 1 つは事前にデータベースに蓄えられたデータを用いて集計値など付加情報を計算し保存しておくことで，問合せ時にその値を用いて集計値を返答する事前処理型方式である．前者には，問合せ時にサンプリングしたデータを用いて近似値を推定し，問合せをインタラクティブに繰り返す毎にその近似解の精度を高めていく方式 [16] がある．この方式では事前処理が不要な為，事前準備の時間や事前に処理した結果を保持するストレージのオーバーヘッドが不要であるというメリットがあるが，問合せ時にサンプリングする処理が追加される為問合せ時間が増大するという問題がある．

後者には，事前にシノプシスと呼ばれる付加情報を計算し保持しておき，それらの情報を用いて問合せ時に近似値を高速に返す方式がある．代表的なものには，全体のデータセットを圧縮した情報を保有する wavelet [17,18] 方式がある．また，データセットの分布を示すヒストグラムを保持して近似解を推定する histogram 方式 [19] がある．この様な事前処理型方式は，オンライン処理型方式に比べ問合せ時の計算コストを抑え近似値の精度が高いというメリットがあるが，事前に付加情報を計算し保持しておくストレージオーバーヘッドが問題である．

上記の様なメリット・デメリットから，オンライン処理型方式と事前処理型方式を組み合わせた近似問合せ方式も提案されている．BlinkDB [20] はサンプリングを事前に実施することで問合せ時のサンプリングコストを軽減させながら，ワークロードを元に定期的に適切なサンプリングを実施する為，より正確な近似値を算出している．ストレージオーバーヘッドを抑制する為，問合せ処理をグルーピングし，保持するサンプルデータを低減させている．[21] らは，シノプシスとサンプリングを組み合わせて，インタラクティブな応答時間で集計値の近似値を返す AQP++を提案しており，[12] らは問合せに応じてどちらを使って近似値を求めるかを決定する方式で高い精度を達成している．この様な研究が実施されているが，特定のワークロードにおいて近似問合せの精度を重視する内容が中心であり，インタラクティブに多様な分析を実施したいユーザニーズに十分応えるものはこれまでの調査では見つかっていない．

6 おわりに

本論文では，多様な問合せに対応する近似問合せを高速に実現する為，オープンソースの関係データベース管理システムである PostgreSQL を利用して，シノプシスと呼ばれる付加情報を埋め込んだ B+Tree 索引を生成し，その索引を利用した検索によって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実装した．初期評価では，PostgreSQL インターフェースを介して集合値の問合せを実行し，オーバーヘッドをほぼ無視できる程度でシノプシスを埋め込んだ索引の生成可能である事と，従来の索引を用いた検索に比べ，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 500 倍以上高速である事を確認した．今後は，多様な問合せクエリに対する近似解処理の優位性の検証や，大規模データを用いた評価を進める．

謝　　辞

本研究の一部は，日本学術振興会科学研究費補助金基盤研究 (B) JP20H04191 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] Surajit Chaudhuri, Bolin Ding, and Srikanth Kandula. Approximate query processing: No silver bullet. In *Proceedings of the 2017 ACM SIGMOD International Conference on Management of Data*, SIGMOD '17, pp. 511–519, New York, NY, USA, May 2017. Association for Computing Machinery.

[5] Kaiyu Li and Guoliang Li. Approximate query processing: What is new and where to go? *Data Sci. Eng.*, Vol. 3, No. 4, pp. 379–397, December 2018.

[6] Hiroki Yuasa, Kazuo Goda, and Masaru Kitsuregawa. Exploiting embedded synopsis for exact and approximate query processing. In *Database and Expert Systems Applications*, pp. 235–240. Springer International Publishing, 2022.

[7] Postgressql. `https://www.postgresql.org/`. 2022-12-27 参照.

[8] Fei Tao, Jiangfeng Cheng, Qinglin Qi, Meng Zhang, He Zhang, and Fangyuan Sui. Digital twin-driven product design, manufacturing and service with big data. *Int. J. Adv. Manuf. Technol.*, Vol. 94, No. 9-12, pp. 3563–3576, February 2018.

[9] Gerald Rebitzer, Tomas Ekvall, Rolf Frischknecht, Davis Hunkeler, Gregory Norris, Tomas Rydberg, W-P Schmidt,

Sangwon Suh, B Pennington Weidema, and David W Pennington. Life cycle assessment part 1: framework, goal and scope definition, inventory analysis, and applications. *Environ. Int.*, Vol. 30, No. 5, pp. 701–720, July 2004.

[10] Postgressql insider. `https://www.postgresql.fastware.com/postgresql-insider-fdw-ove`. 2022-12-27 参照.

[11] Tpc-h. `https://www.tpc.org/tpch/`. 2022-12-27 参照.

[12] Xi Liang, Stavros Sintos, Zechao Shang, and Sanjay Krishnan. Combining aggregation and sampling (nearly) optimally for approximate query processing. In *Proceedings of the 2021 ACM SIGMOD International Conference on Management of Data*, SIGMOD '21, pp. 1129–1141, New York, NY, USA, June 2021. Association for Computing Machinery.

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

[14] Jim Gray, Surajit Chaudhuri, Adam Bosworth, Andrew Layman, Don Reichart, Murali Venkatrao, Frank Pellow, and Hamid Pirahesh. Data cube: A relational aggregation operator generalizing Group-By, Cross-Tab, and Sub-Totals. *Data Min. Knowl. Discov.*, Vol. 1, No. 1, pp. 29–53, March 1997.

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

[16] 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, August 1999.

[17] Ioannis Mytilinis, Dimitrios Tsoumakos, and Nectarios Koziris. Distributed wavelet thresholding for maximum error metrics. In *Proceedings of the 2016 ACM SIGMOD International Conference on Management of Data*, SIGMOD '16, pp. 663–677, New York, NY, USA, June 2016. Association for Computing Machinery.

[18] Abdul Naser Sazish and Abbes Amira. An efficient architecture for HWT using sparse matrix factorisation and DA principles. In *APCCAS 2008 - 2008 IEEE Asia Pacific Conference on Circuits and Systems*, pp. 1308–1311. ieeexplore.ieee.org, November 2008.

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

[20] 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 *Proceedings of the 8th ACM European Conference on Computer Systems*, EuroSys '13, pp. 29–42, New York, NY, USA, April 2013. Association for Computing Machinery.

[21] Jinglin Peng, Dongxiang Zhang, Jiannan Wang, and Jian Pei. AQP++: Connecting approximate query processing with aggregate precomputation for interactive analytics. In *Proceedings of the 2018 ACM SIGMOD International Conference on Management of Data*, SIGMOD '18, pp. 1477–1492, New York, NY, USA, May 2018. Association for Computing Machinery.