$(PC)^2L$ におけるビッグデータ可視化分析を想定したデータ処理最適化のための SQL クエリ抽出

稲澤　朋也†　　能條　太悟††　　富井　尚志†††

† 横浜国立大学理工学部数物・電子情報系学科　〒 240–8501 横浜市保土ヶ谷区常盤台 79–5
†† 横浜国立大学大学院環境情報学府情報環境専攻　〒 240–8501 横浜市保土ヶ谷区常盤台 79–7
††† 横浜国立大学大学院環境情報研究院　〒 240–8501 横浜市保土ヶ谷区常盤台 79–7
E-mail: †inazawa-tomoya-xy@ynu.jp, ††nojo-daigo-gf@ynu.jp, †††tommy@ynu.ac.jp

あらまし　我々は PCP（Parallel Coordinates Plot）に基づく多変量データ解析支援システム（(PC)^{2}DV: Parallel Coordinates Plot Commutative Data Visualizer）を提案してきた．(PC)^{2}DV では SQL に類似した言語（(PC)^{2}L: Parallel Coordinates Plot Commutative Language）によってデータ操作とデータ可視化が可能である．しかしながら，データ操作におけるフロントエンドでの独自処理が非効率であるために，ビッグデータ解析が困難であることが欠点だった．本研究では，データ操作処理の際にデータベース管理システム（DBMS）による最適化が行われるよう拡張する．具体的にはまず，(PC)^{2}L をデータ操作部とデータ可視化部に分け，データ操作部を SQL に変換する．次に，その SQL をバックエンドの DBMS で実行する．これにより DBMS の最適化機構を利用したデータ操作処理が可能となる．データ操作処理時間に関する評価実験により提案システムの有効性を示す．

キーワード　情報可視化，平行座標プロット PCP，多変量データ解析，データ操作，ビッグデータ

1 はじめに

近年，センサ技術やストレージ技術の発達により，属性数の多い多変量データを大量に取得・蓄積することが可能になった．それに伴い，大量の多変量データの利活用を目的としたデータ分析の需要が高まっている．

多変量データ解析を支援するため，我々は (PC)^{2}DV（Parallel Coordinates Plot Commutative Data Visualizer）を提案してきた [1–3]．(PC)^{2}DV は，可視化された多変量データに対してアドホックなクエリが実行できる GUI を有する汎用のミドルウェアである．外部からロードしてきた多変量データを可視化し，SPJ（Selection-Projection-Join）質問のような関係代数演算が表現可能な GUI として PCP（Parallel Coordinates Plot）[4, 5] を利用する．ユーザはそれらの操作に加え，軸の配置変更（Coordinating）や任意の軸の属性値を用いた色分け（Coloring）といった操作によりデータ分析を進める．また，(PC)^{2}DV には，PCP 上のインタラクションと可視化の状態を保存・再現する SQL に類似した (PC)^{2}L（Parallel Coordinates Plot Commutative Language）という言語が組み込まれている．(PC)^{2}L の記述を (PC)^{2}DV 内のロードデータに対して適用することで，関係代数演算における SPJ 質問に相当するデータ操作や任意のグラフによる可視化が可能となる．

(PC)^{2}L は，(1) 言語解析，(2) データ操作処理，(3) 描画処理の手順によってフロントエンドで独自に処理される．ところで，(PC)^{2}DV 内で扱われるデータはインデックスを用いたデータ構造になっていないため，(2) で全探索の必要性が生じる場合がある．データ件数が増えるにつれデータ処理時間が線形時間で増大するため，ビッグデータ処理が困難となる．

そこで本研究では，(PC)^{2}DV におけるビッグデータ分析を想定して，(2) でデータベース管理システム（DBMS）の最適化機構を利用したデータ操作処理が行われるよう拡張する．そのためにまず，(PC)^{2}L から SQL を抽出することを行う．(PC)^{2}L はデータ操作部とデータ可視化部に分けられ，データ操作部は言語仕様上 SQL と可換である．(1) の結果を基に (PC)^{2}L のデータ操作部を SQL に変換する．次に，ユーザ指定の DBMS に ODBC（Open Database Connectivity）接続してその SQL を実行する．DBMS から実行結果データを受け取ったら (3) に移る．

これによりデータ操作処理は常に DBMS により最適化される．また，ユーザは DBMS 上のデータに対して (PC)^{2}DV を介した可視化分析を行うため，データロードが不要となる．データ操作処理時間に関する従来システムとの比較実験により，提案システムの有効性を示す．

2 関連研究

2.1 PCP とデータ操作

PCP は 1985 年，Inselberg によって初めて概念が定義された [4]．それ以降，PCP に関する様々な議論がなされており，情報可視化の分野において重要なトピックの 1 つとなっている．Johansson らによれば，PCP の研究カテゴリーは次の 4 つに分類される [5]: （1）PCP の軸のレイアウト, (2）PCP の乱雑さ（Clutter）の軽減方法, (3）PCP の実用性の提示, (4）PCP とほかのデータ解析手法との比較．上記の通り，PCP の見せ方に関する議論がほとんどであり，PCP の操作に着目した議

論はされていない．また，PCP の可視化と SPJ 質問をかけ合わせるような議論はされていない．

また，Bouali らは，対話型遺伝的アルゴリズムを使用して可視化手法を推薦するシステムを構築した [6]．これは，データや利用者の要求に応じてより適切な可視化手法（散布図行列や PCP など）の選択を支援するものである．我々の提案する $(PC)^2$DV は関係代数における SPJ 質問に相当する表現力を持つような可視化システムを構築するため，タプルが 1 つの線で明示され，詳細に参照・分析可能である PCP が適切である．

一方で，インタラクティブに操作しながら PCP による分析を支援するシステムの提案もされている．Itoh らは，属性軸間の相関に基づいてインタラクティブに次元削減を行い，PCP から所望する情報の発見を支援するシステムを構築した [7]．Zhou らは，エントロピーの概念を導入することで，クラスタに基づいて PCP の属性軸の整列順序を決定する手法の提案をした [8]．

また，多変量データを可視化するその他の手法として，複数の散布図を表示する散布図行列があげられる [9]．散布図行列は，属性同士の相関を直感的に把握できる一方，散布図数が属性数の 2 乗に比例して増加する．そのため，属性数が多いデータでは非常に大きい画面空間を使う必要があり，データ操作の過程で SPJ 質問（特に Join）を適用することは不向きであるといえる．

2.2 データ解析支援

データやシステムの操作過程を管理する研究（Provenance）が行われている [10]．特にデータやシステム，プログラミングコードなどの操作過程や意図を保存することは，複雑なデータ解析を支援するために重要であるといわれている．さらに．分析データの操作の過程や意図は，SQL のような関係代数演算をサポートする言語の記述により示すことが有効であるといわれている．この点において，SQL に類似する言語である $(PC)^2$L を用いて $(PC)^2$DV におけるデータの操作過程を保存することは有効な手段であるといえる．

また，データやシステムの操作過程を保存することでユーザの支援を行う手法の提案がなされている．Waldner らは，PC のアプリケーション操作ログを保存し，それらを時系列が理解できるように可視化することで，ユーザが過去に行った情報探索の詳細を再現する支援を行った [11]．Mindek らは，画像データと分析過程に利用する他のデータソースのデータを同時に表示し，分析者の文脈を保存したスナップショットを保存することで，シミュレーションデータの可視化や文書分析の支援を行った [12]．Gratzl らは，様々な可視化手法を組み合わせて複数のデータソースから得られたデータとその分析過程を可視化し，データ分析の支援を行った [13]．これらの手法と比較して我々の手法は,「可視化システムのデータ分析過程を可視化して見せる」のではなく,「SQL に類似した言語を用いてデータ分析の途中経過を保存し，問合せ言語として一般的な SQL に親しみのあるデータ分析者を支援する」ものであり，立場が異なる．また，言語を用いて操作過程を保存することで，言語の一部を書き換えるだけで容易にデータ分析の改善をすることができる．その点でこれらの研究と比較して優位性をもつ．

また，大量のデータを対象として，インタラクティブにデータ可視化を行う研究が行われている [14]．中でも，関係データベーススキーマに基づくデータに対し，GUI 上でクエリの記述や複数の可視化の連携を可能にし，データ解析を支援する研究も複数行われている．Derthick らは，データオブジェクトを可視化しながら GUI でクエリが表現可能な環境を構築した [15]．North らは，データの可視化と，表示した複数の可視化間の連携をユーザーが自由に変更可能なインターフェースの構築を行った [16]．杉渕らは，クエリフローモデルによる直感的かつ段階的なクエリが構築可能な GUI を機能として備えた可視化フレームワークを実装した [17]．これらの研究は，可視化とクエリを GUI 上で連携させることで，インタラクティブなデータ解析を支援する点では，我々と立場が同じと言える．しかしこれらの研究は,「データベースに習熟していないデータ解析者を支援する」点を重視している．本研究は,「データ解析過程と可換な SQL に類似した言語により，データベースや SQL に習熟した解析者を支援する」点で，これらの研究とは立場が異なる．

3 $(PC)^2$DV の概要と内部処理

本章では，3.1 節で $(PC)^2$DV の概要を述べた後，$(PC)^2$DV の構成要素のうち，本研究による拡張に関連する部分であるデータロード部と $(PC)^2$L の内部処理について，3.2 節，3.3 節で述べる．

3.1 $(PC)^2$DV の概要

$(PC)^2$DV では，データ分析者が多変量データを可視化する PCP に対してインタラクションを行いつつ，その操作結果を $(PC)^2$L で保存・再現する．それにより，試行錯誤を行いながら所望の可視化を獲得することを想定する．以下では，想定する操作手順の流れを示す．

（1） 分析対象のデータをリレーションとしてロードする．

（2） ロードしたリレーションを PCP により可視化する．

（3） PCP を補完する形で，任意のグラフで描画を行う．

（4） 可視化結果を基に，PCP 上のインタラクションまたは $(PC)^2$L の記述によりデータ操作を行う．その際，データ操作結果をリアルタイムに（3）で表示したグラフに反映する．

（5） データ分析者が所望するときに,（3）,（4）の分析過程のスナップショットを $(PC)^2$L で保存する．

（6） （3）-（5）を繰り返す．その際，過去のスナップショットに戻る必要がある場合，該当する $(PC)^2$L を入力してシステム上に分析過程を再出力する．

（7） データ分析者が所望の可視化結果を獲得する．

$(PC)^2$DV の表示例を図 1 に示す．

図 1 中の A:PCP View では，ロードデータを可視化した PCP が表示される．ここでは，先行研究 [1] で定義したデータに対するインタラクションのうち，選択（Selection），色分け（Coloring），軸配置（Coordinating）の操作が利用可能であ

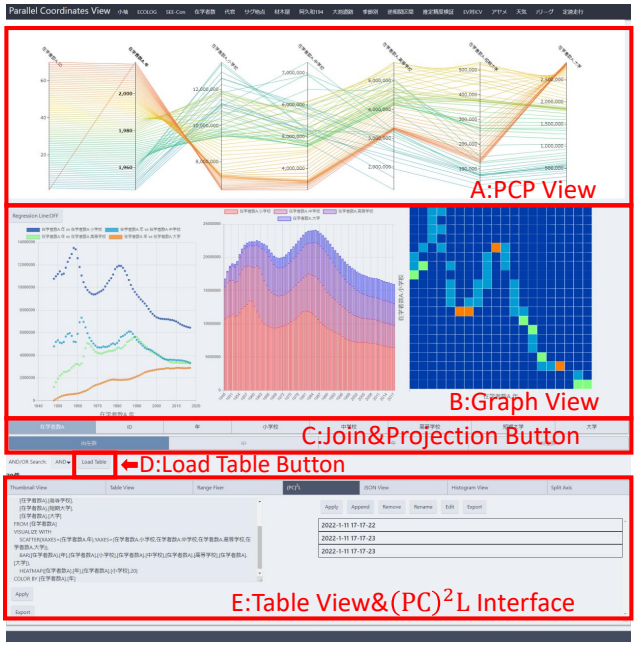

図 1: (PC)²DV の表示例

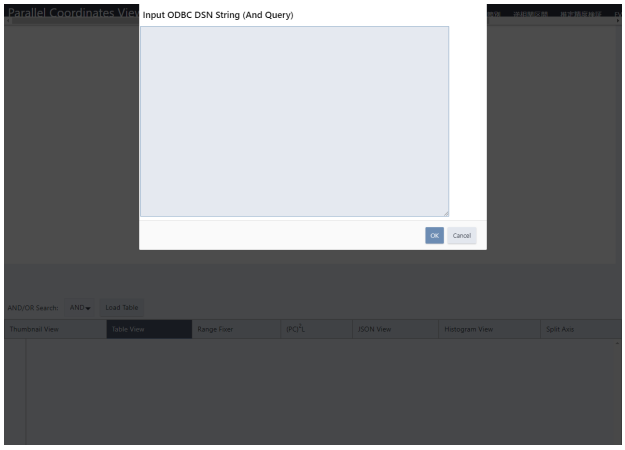

図 2: データロードに必要な情報を入力する画面

る．PCP の各軸上を上下にドラッグし範囲選択をすることで，範囲内に含まれる線のみが PCP に表示される．これにより選択（Selection）の操作が可能である．なお，本システムで利用可能な集約関数の軸については，他の軸の選択（Selection）操作結果がリアルタイムに集約値に反映される．

図 1 中の B:Graph View では，(PC)²L で指定したグラフが表示される．A，C 上で行ったデータ操作結果はリアルタイムにグラフ上に反映される．

図 1 中の C:Join&Projection Button では，横一行が 1 つのリレーションに対応するトグルボタンが表示される．ここでは，先行研究 [1] で定義したデータに対するインタラクションのうち，結合（Join）と射影（Projection）が利用可能である．

図 1 中の D:Load Table Button では，(PC)²DV の外部のデータソースにあるデータを取得する．Load Table Button を押すと，外部のデータソースへの接続に必要な情報を入力する画面が表示される（図 2）．

図 1 中の E:Table View&(PC)²L Interface では，A，C 上で行ったインタラクションを反映したデータセットのテーブル表示と，(PC)²L の入出力を受け付けるインターフェースを持つ．

3.2 (PC)²DV のデータロード部とその内部処理

データロード部は，(PC)²DV の外部のデータソースにあるデータを取得する機能を持つ．すなわち，上記に述べたデータ操作手順の（1）を担う部分である．外部のデータソースへのアクセスには ODBC を用いる．そのため，データソースの形式が ODBC 接続可能な形式であれば連想配列としてデータの取得が可能であり，(PC)²DV の分析対象にすることができる．図 2 の画面上に記述されたデータ接続のための DSN クエリ，接続したデータソースに対する問合せのための SQL クエリを受理し，ODBC 接続によるデータ取得を実行する．

先行研究での内部実装として，ロードされた全データは JSON 構造のデータとしてブラウザ上のメモリに保持される．保持できるデータ量はブラウザに割り当てられるメモリに依存するため，ビッグデータをロードしようとするとメモリ不足になる場合がある．

3.3 (PC)²L とその内部処理

(PC)²L は，PCP 上のインタラクションの状態を保存・再現可能な言語表現である．この言語記述を (PC)²DV 上のロードデータに対して適用することにより，データ操作や任意のグラフによる可視化が可能となる．本言語では，SQL の SELECT 句，FROM 句，JOIN 句，WHERE 句，GROUP BY 句，HAVING 句に加えて，COLOR BY 句，VISUALIZE WITH 句が利用できる．前五者はデータ操作，後二者はデータ可視化に対応し，以後それぞれをデータ操作部，データ可視化部と呼ぶ．また，本言語では以下に示す集約演算が表現できる．

- 合計（SUM）
- 平均（AVG）
- 最大値（MAX）
- 最小値（MIN）
- データ件数（COUNT）
- 中央値（MED）
- 四分位数（QUARTILE）

(PC)²L は次の手順によって内部で処理を行う．まず，受理した (PC)²L を構文解析器により抽象構文木（AST）に変換する言語処理を行った後，データ操作処理を行う．次に，その AST とデータ操作処理結果を基に PCP や任意のグラフによる描画を行う．

データ操作処理とは，分析対象データに対する操作における内部処理であり，フロントエンドで独自に処理する．ところで，分析対象の多変量データは連想配列であり，インデックスが用いられたデータ構造でないため，例えば集約値を求める集約処理に全探索を要する．全探索に起因して，データ件数が増えるにつれデータ操作処理時間は線形時間で増大する．

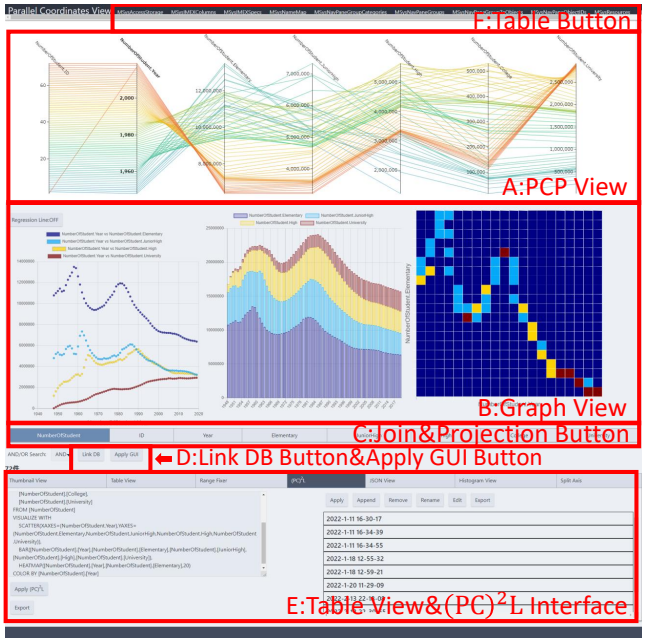

図 3: 拡張を行った (PC)²DV の表示例

4　データ操作処理に関するシステム拡張

4.1　データ操作処理改善の概要

3.3 節で述べたように，従来の (PC)²DV では，(PC)²L のデータ操作におけるフロントエンドでの独自処理が非効率だった．(PC)²DV で分析対象となる多変量データは，データ件数の多いビッグデータであることが想定される．分析対象がビッグデータである場合，フロントエンドで毎回のデータ操作処理に長時間を要するため，スムーズなデータ分析を支援することができない．そこで，本研究では (PC)²L のデータ操作処理において，常にリンク先 DBMS の最適化機構を利用した処理が行われるよう拡張を行った．ここで,「リンク」とは「指定のデータソースに対して常に ODBC 接続が可能な状態」と定義する．本拡張により，データ分析者は (PC)²DV にロードされたデータではなく，リンク先 DBMS 内のデータに対して直接 (PC)²L を記述する．(PC)²DV は，(PC)²L を受理する度にデータ操作部を SQL に変換し，リアルタイムにその SQL をリンク先 DBMS で実行する．その際，リンク先 DBMS により最適化機構を利用したデータ操作処理が行われる．リンク先 DBMS から取得した結果データと (PC)²L のデータ可視化部を基に描画処理を行うことで，ビッグデータを想定した多変量データの可視化分析を支援する．本拡張は 4.2 節で後述する (PC)²L からの SQL クエリ抽出，4.3 節で後述する DBMS リンクにより実現される．拡張を行った (PC)²DV の表示例を図 3 に示す．図 3 中の A，B，C，E は，従来システム（図 1）の A，B，C，E と等しい．図 3 中の D:Link DB Button&Apply GUI Button，F:Table Button については 4.3 節で述べる．なお，本拡張により，従来システムで表現可能だった集約関数のうち MED と QUARTILE のサポートは失われる．

表 1: SELECT 句で利用可能なオプション

説明	記法	変数
軸範囲の設定	RANGE $= A$ BETWEEN y_1 AND y_2	A: 範囲を参照する対象の軸名 y_1: 軸の最小値 y_2: 軸の最大値
軸名の設定	AS $alias$	$alias$: 任意の軸名

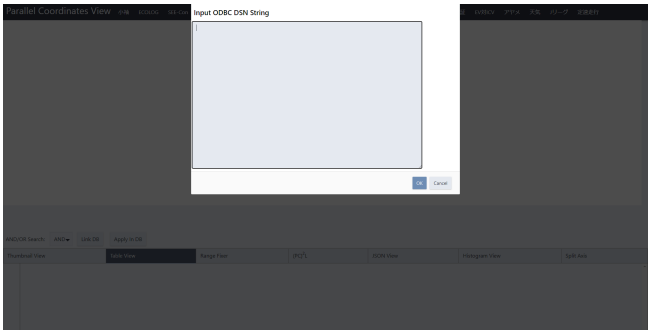

図 4: リンクに必要な DSN クエリを入力する画面

4.2　SQL クエリ抽出

本節では，拡張の 1 つ目である SQL クエリ抽出について述べる．SQL クエリ抽出では，受理した (PC)²L から DBMS で実行したい SQL を生成する．具体的には，(PC)²L の原文と (PC)²L の言語処理結果である AST を基に，(PC)²L のデータ操作部を SQL に変換する．

(PC)²L のデータ操作部のうち，SELECT 句以外の FROM 句，JOIN 句，WHERE 句，GROUP BY 句，HAVING 句については，定義上の文法が SQL と全く等しい．SELECT 句については,「< SELECT 句> <オプション>」と記述することで<オプション>に指定した機能が利用可能である（表 1）．これらは (PC)²L で独自に定義した句や，DBMS でデータ操作を実行する際に無関係な句であるため，SQL 変換時に排除する必要がある．

以上のことから，以下の手順により SQL を生成する．まず，AST を基にしてオプションを排除した SELECT 句を再構築する．次に，原文から切り出した FROM 句，JOIN 句，WHERE 句，GROUP BY 句，HAVING 句を利用する．

4.3　DBMS リンク

本節では，拡張の 2 つ目である DBMS リンクについて述べる．DBMS リンクでは，(PC)²L によるデータ操作がリアルタイムにリンク先 DBMS で実行されるようシステムを拡張する．

本拡張により，図 3 中 D の Link DB Button を押した後，DSN（Data Source Name）クエリの入力を受け付ける画面が表示される（図 4）．データ分析者は，分析開始時に図 4 の画面上で，(PC)²DV のリンク先として任意の DBMS を指定するための DSN クエリを記述する．(PC)²DV は，受理した DSN クエリを基に ODBC 接続を実行し，接続成功時は DSN クエリが保持され，リンクが完了する．失敗時は再度 DSN クエリの入力を促す．リンク完了に伴い図 4 の画面が閉じ，リンク先 DBMS が有する全てのデータベーステーブルが図 3 中の F:Table Button に表示される．テーブル名をクリックすると

図 3 中の C が表示される．

リンク完了後，データ分析者は，リンク先 DBMS 内のデータに対し直接 $(PC)^2L$ を記述する．その際，$(PC)^2DV$ は受理した $(PC)^2L$ のデータ操作部を，4.2 節の処理により SQL に変換し，リアルタイムにリンク先 DBMS で実行する．DBMS から取得した実行結果と $(PC)^2L$ のデータ可視化部を基に描画処理を行う．

$(PC)^2L$ のデータ操作処理結果を可視化する PCP に対し，データ分析者が PCP 上で選択（Selection）操作を行うことが想定される．その際，集約関数の軸については，他の軸の選択（Selection）の操作結果に応じて集約値が変化する．しかしながら，本拡張により $(PC)^2DV$ は集約処理をせず，他の軸の選択（Selection）の操作結果を集約値に反映することができないため，集約関数の軸のデータ可視化は中止される．選択（Selection）の操作結果を集約演算に反映したい場合は，図 3 中 D の Apply GUI Button をクリックすることにより達成される．これはクリック操作に応じて PCP の状態が $(PC)^2L$ に変換され，$(PC)^2L$ を実行した際と同様に処理されるためである．また，図 3 中 C による射影（Projection）・結合（Join）の操作についても，本拡張により分析対象データをフロントエンドに持たない $(PC)^2DV$ は処理不可能であるため，操作結果をリアルタイムに PCP に反映することができない．射影（Projection）・結合（Join）の操作結果を PCP に反映したい場合も，図 3 中 D の Apply GUI Button をクリックすることにより達成される．

以上のことから，提案システムにおいて想定するデータ分析者の操作手順の流れを以下に示す．

（1） 任意の DBMS へリンクする．

（2） $(PC)^2L$ を記述し，データ操作ならびにデータ可視化を行う．

（3） 可視化結果を基に，GUI 上のインタラクションを行う．その際，データ操作結果を（3）で表示したグラフに反映する．

（4） データ分析者が所望するときに，(2) , (3）の解析過程のスナップショットを $(PC)^2L$ で保存する．

（5） (2) - (4）を繰り返す．その際，過去のスナップショットに戻る必要がある場合，該当する $(PC)^2L$ を入力してシステム上に分析過程を再出力する．

（6） データ分析者が所望の可視化結果を獲得する．

4.4 クエリ変換の例

本節では，実際に $(PC)^2L$ のクエリ文から具体的にどのような SQL 文が抽出され，どのように $(PC)^2DV$ で実行結果が可視化されるかを示す．本例では，日本の気象庁のホームページ [1]から取得した日毎の最高気温と最低気温のデータを分析対象データとする．なお，取得した気象データは 2021 年 1 月 1 日から 2021 年 12 月 31 日の神奈川県横浜市における，日毎の最高気温と最低気温であり，データ件数は 365 件である．実験環境の MS ACCESS でデータベース test.accdb を作成し，test.accdb

1：https://www.data.jma.go.jp/gmd/risk/obsdl/index.php

表 2: Temperature の属性

属性	説明
Year	年
Month	月
Day	日
TemperatureMax	日毎の最高気温
TemperatureMin	日毎の最低気温

```
Driver={Microsoft Access Driver (*.mdb, *.accdb)};Dbq=C:¥test.accdb;
```

図 5: ”test.accdb”接続のための DSN クエリ

```
SELECT    Temperature.Month
              AS 月,
          AVG(Temperature.MaxTemperature)
              BETWEEN 0 AND 35
              AS 最高気温の平均,
          AVG(Temperature.MinTemperature)
              BETWEEN 0 AND 35
              AS 最低気温の平均
FROM      Temperature
VISUALIZE WITH
      LINE(Temperature.Month, AVG(Temperature.MaxTemperature),
      AVG(Temperature.MinTemperature))
GROUP BY      Temperature.Month
COLOR BY       Temperature.Month
```

図 6: ”test.accdb”に対する $(PC)^2L$

```
SELECT    Temperature.Month,
          AVG(Temperature.MaxTemperature),
          AVG(Temperature.MinTemperature)
FROM      Temperature
GROUP BY      Temperature.Month
```

図 7: ”test.accdb”に対する $(PC)^2L$ から抽出された SQL

内に取得した気象データをテーブル Temperature として保存した．Temperature が有する属性を表 2 に示す．test.accdb にリンクした後，Temperature 内のデータに対し 2021 年の月毎の最高気温と最低気温の平均を求め，可視化する．

test.accdb へのリンク　test.accdb へのリンクに際し，図 4 の画面上に記述した DSN クエリを図 5 に示す．DSN クエリには MS ACCESS の ODBC ドライバ，”test.accdb”のディレクトリを指定した．

$(PC)^2L$ 記述によるデータ操作とデータ可視化　記述した $(PC)^2L$ を図 6 に示す． 月毎の最高気温と最低気温の平均を求めるために，GROUP BY 句および集約関数を用いた．具体的には，GROUP BY 句により月の値でグループ化し，集約関数の AVG により最高気温と最低気温の平均を求めた．なお，PCP の見やすさのために，SELECT 句のオプション機能である BETWEEN 句による軸範囲と AS 句による軸名の設定，COLOR BY 句を用いた月の軸による色分けも同時に行った．また，VISUALIZE WITH 句を用いて，折れ線グラフで月毎の最高気温と最低気温の平均の変化を可視化した．記述した $(PC)^2L$ を実行後，抽出された SQL 文を図 7 に示す．このように，SELECT 句のオプション機能及びデータ可視化部を除いた記述部分が SQL として抽出される．また，$(PC)^2L$ の実行結果として描画された PCP を図 8 に，折れ線グラフを図 9 に示す．DBMS から取得

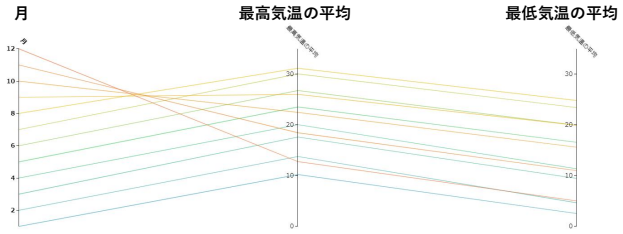

図 8: ”test.accdb”に対する $(PC)^2L$ の実行結果 PCP

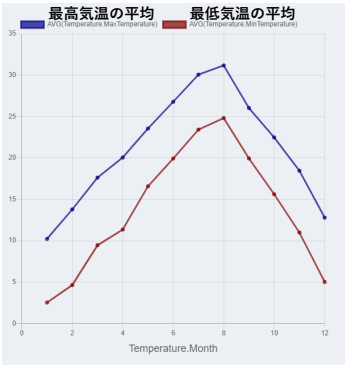

図 9: ”test.accdb”に対する $(PC)^2L$ の実行結果折れ線グラフ

した SQL の実行結果データは PCP で表示される折れ線に対応する．

5 評価実験

4 章で述べた通り，本研究では $(PC)^2DV$ のデータ操作処理時間に関するシステム拡張を行った．本章では，評価実験により提案システムのビッグデータ分析への有効性を示す．

5.1 実験概要

本実験では，データ件数の異なるデータセット毎に，(1) 従来のシステムを用いた場合，(2)DBMS 内の分析対象データにインデックスの設定がなされていない状態で提案システムを用いた場合，(3)DBMS 内の分析対象データにインデックスの設定がなされている状態で提案システムを用いた場合，それぞれで $(PC)^2L$ のデータ操作処理時間を計測した．データ操作処理時間は，受理した $(PC)^2L$ の，言語処理終了時点から描画処理開始時点までと定義する．提案システムを用いた場合を (2) と (3) に分ける意図は，リンク先 DBMS の最適化機構が機能しない場合と機能する場合で提案システムの有効性が変わることを明示するためである．$(PC)^2L$ が異なる 3 つの実験で処理時間を計測し，従来システムと提案システムを比較評価した．

実験用データセット MS ACCESS で実験用のデータベースを作成し，データ件数を N（100 | 1,000 | 10,000 | 100,000 | 1,000,000 | 10,000,000）として，表 3 に示す属性を有するテーブルを 6 つ作成した．また SQL Server に作成した実験用データベースにも同様のテーブルを，データ件数を N（100 | 1,000 | 10,000 | 100,000 | 1,000,000 | 10,000,000 | 100,000,000）として 7 つ作成した．なお，(3) におけるインデックス設定については，両者とも id と remainder の両方に B-tree インデックスを貼り，id は重複を許さない設定とした．

表 3: 実験用データベーステーブルの属性

属性	説明
id	[0,N) の連続整数
remainder	id を 10 で割った剰余

```
SELECT [TABLE].remainder, MAX([TABLE].id)
FROM  [TABLE]
GROUP BY  [TABLE].remainder
COLOR BY  [TABLE].id
```

図 10: GROUP BY 句及び集約演算を含む $(PC)^2L$

```
SELECT [TABLE].remainder, MAX([TABLE].id)
FROM  [TABLE]
WHERE ([TABLE].id BETWEEN 2*<NUM_OF_RECORDS>/10 AND 8*<NUM_OF_RECORDS>/10)
        AND
        ([TABLE].remainder BETWEEN 2 AND 8)
GROUP BY  [TABLE].remainder
COLOR BY  [TABLE].id
```

図 11: GROUP BY 句及び集約演算，範囲質問を含む $(PC)^2L$

```
SELECT MAX([TABLE].id), MAX([TABLE].remainder)
FROM  [TABLE]
COLOR BY  [TABLE].id
```

図 12: 集約演算を含む $(PC)^2L$

実験 1 実験 1 では，前述した MS ACCESS の実験用データセットに対して，GROUP BY 句及び集約演算を含む $(PC)^2L$ を実行した際のデータ操作処理時間を計測した．テーブル名を一般化したクエリを図 10 に示す．

実験 2 実験 2 では，実験 1 と同じく MS ACCESS の実験用データセットに対して，実験 1 で使用した $(PC)^2L$ に範囲質問を追加した $(PC)^2L$ を実行し，データ操作処理時間を計測した．テーブル名及び WHERE 句の条件式を一般化したクエリを図 11 に示す．

実験 3 実験 3 では，前述した SQL Server の実験用データセットに対して，集約演算を含む $(PC)^2L$ を実行した際のデータ操作処理時間を計測した．テーブル名を一般化したクエリを図 12 に示す．ところで，SQL Server では発行されたクエリに対する実行プランが確認可能である．$(PC)^2L$ から抽出される SQL（COLOR BY 句を除いた記述部分）に対する SQL Server 側の実行プランは，(2) では概ね全探索（図 13），(3) ではインデックススキャン（図 14）であることが確認された．

5.2 実験結果と考察

実験 1 実験結果を表 4 と図 15 に，考察を以下に示す．

- 従来システムにおいて，100,000 件を超えるデータ件数ではフロントエンドへのデータロードが不可能となった．3.2 節に述べたように，メモリ不足になったと判断される．
- データ件数の増加に伴い，[(1) のデータ操作処理時間] > [(2) 及び (3) のデータ操作処理時間] となり．データ操作処理時間の大幅な改善が示された．例えば，データ件数が 100,000 件の際，処理時間は従来システムでは約 100 秒，提案システム

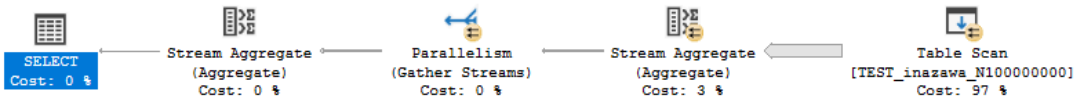

図 13: 実験 3 における (2) の SQL Server 側のクエリ実行プラン

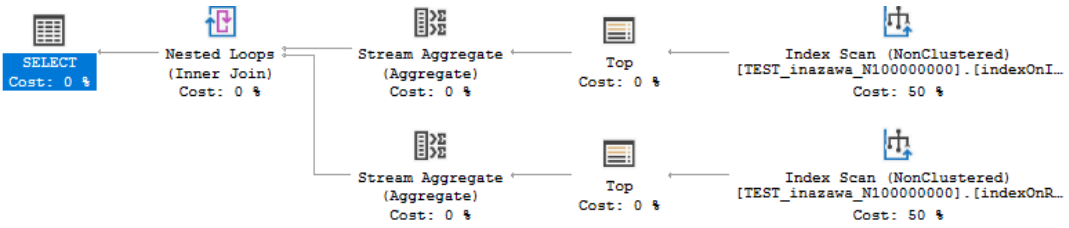

図 14: 実験 3 における (3) の SQL Server 側のクエリ実行プラン

表 4: 実験 1 におけるデータ操作処理時間の計測結果

データ件数	データ操作処理時間 [ms]		
	従来システム	提案システム	
		インデックス無	インデックス有
100	64	320	317
1,000	86	317	310
10,000	1060	327	329
100,000	93651	436	435
1,000,000		1331	1298
10,000,000		10398	10337

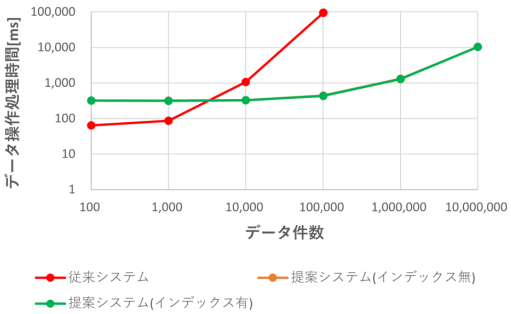

図 15: 実験 1 におけるデータ操作処理時間の計測結果

表 5: 実験 2 におけるデータ操作処理時間の計測結果

データ件数	データ操作処理時間 [ms]		
	従来システム	提案システム	
		インデックス無	インデックス有
100	51	339	310
1,000	76	312	311
10,000	1707	331	313
100,000	156383	406	320
1,000,000		1218	327
10,000,000		9306	827

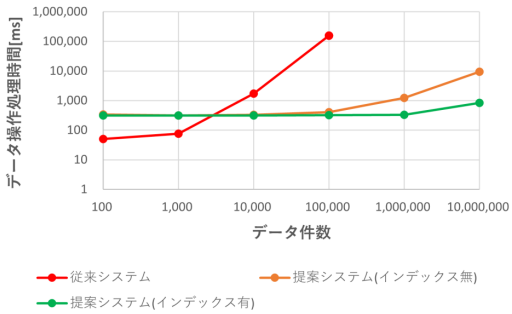

図 16: 実験 2 におけるデータ操作処理時間の計測結果

では約 0.4 秒となった．しかしながら，提案システムにおいて，インデックス設定による処理時間の向上が認められなかった．すなわち，DBMS の最適化機構の利用に関わらず，提案システムの方がより優れたデータ操作処理が行われた．ゆえに，この改善は処理機構の違いに依るものと考えられる．なお，最適化機構が機能しなかったのは，インデックスは最大値の検索には有効だが，その有効性が GROUP BY 句によるグループ化により失われたからだと推察される．

- データ件数が少ない場合では，[(1) のデータ操作処理時間] < [(2) 及び (3) のデータ操作処理時間] となった．これは，提案システムのデータ操作処理時間において，本質的となる DBMS 側のデータ操作処理よりもデータ接続及びデータ取得に要する時間の方が影響が大きかったからだと判断される．

実験 2　実験結果を表 5 と図 16 に，考察を以下に示す．

- 提案システムにおいて，DBMS 上でのインデックス設定による処理時間の向上が認められた．具体的には，10,000,000 件のデータに対する (2) のデータ操作処理時間は約 10 秒，(3) のデータ操作処理時間は約 1 秒という結果となった．実験 1 の結果を踏まえ，これは選択処理にインデックスが効いたからだと判断される．また，この実験結果により提案システムのビッグデータへの有効性が示された．
- その他従来システムにおいて，実験 1 と同様な結果が得られた．

実験 3　実験結果を表 6 と図 17 に，考察を以下に示す．

- 提案システムにおいて，データ件数の増加に関わらず (3) のデータ操作処理時間は約 0.2 秒という結果となった．SQL

表 6: 実験 3 におけるデータ操作処理時間の計測結果

データ件数	データ操作処理時間 [ms]		
	従来システム	提案システム	
		インデックス無	インデックス有
100	41	159	187
1,000	56	147	165
10,000	1152	179	173
100,000	95039	209	138
1,000,000		305	177
10,000,000		351	193
100,000,000		1582	187

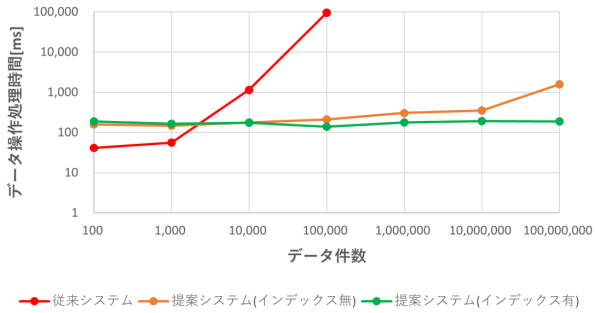

図 17: 実験 3 におけるデータ操作処理時間の計測結果

Server 側では図 14 に示されたプランでインデックスに基づく集約処理が行われ，取得するデータ量はデータ件数に依らず 10 レコードであるから，妥当な結果だといえる．また実験 2 と同様，提案システムのビッグデータへの有効性が示された．

- その他従来システムにおいて，実験 1 と同様の結果が得られた．

6 まとめと今後の課題

本研究では，$(PC)^2L$ のデータ操作処理に着目し，先行研究による実装ではビッグデータ対応が困難だったことを踏まえ，改善を行った．具体的には，データ操作の際にフロントエンドで独自処理するのではなく，SQL 抽出と DBMS リンクによりバックエンドの DBMS に最適処理させるようシステムを拡張した．本拡張により $(PC)^2DV$ は，$(PC)^2L$ による DBMS 内データの操作結果をリアルタイムに可視化することで，データの可視化分析を支援する．また，データ操作処理時間に関する従来システムと提案システムの比較実験を行い，提案システムのビッグデータ分析への有効性を示した．

今後の課題として，ビッグデータ可視化支援のための $(PC)^2L$ と GUI の拡張などが挙げられる．

謝辞 本研究の一部は横浜国立大学令和 3 年度学長戦略経費の支援による．

文　献

[1] 濱崎裕太, 植村智明, 富井尚志. 多変量データを SPJ 質問により統合する平行座標プロット型情報可視化システムと操作言語. 情報処理学会論文誌データベース（TOD）, Vol. 12, No. 4, pp. 27–39, October 2019.

[2] 植村智明, 吉田顕策, 吉瀬雄大, 富井尚志. 試行錯誤を許容するデータ解析支援システムと電気自動車の走行ログ解析. 情報処理学会論文誌データベース（TOD）, Vol. 13, No. 4, pp. 13–26, October 2020.

[3] 植村智明, 能條太悟, 吉瀬雄大, 富井尚志. 解析者の興味に基づく道路区間集計が可能な EV 推定消費エネルギーデータ解析システムの構築と応用. 情報処理学会論文誌データベース（TOD）, Vol. 14, No. 4, pp. 70–85, October 2021.

[4] Alfred Inselberg. The plane with parallel coordinates. *The Visual Computer*, Vol. 1, No. 2, pp. 69–91, 1985.

[5] Jimmy Johansson and Camilla Forsell. Evaluation of parallel coordinates: Overview, categorization and guidelines for future research. *IEEE Trans. on Visualization and Computer Graphics (TVCG)*, Vol. 22, No. 1, pp. 579–588, 2016.

[6] Fatma Bouali, Abdelheq Guettala, and Gilles Venturini. VizAssist: An interactive user assistant for visual data mining. *The Visual Computer: Int'l Journal of Computer Graphics*, Vol. 32, No. 11, pp. 1447–1463, 2016.

[7] Takayuki Itoh, Ashnil Kumar, Karsten Klein, and Jinman Kim. High-dimensional data visualization by interactive construction of low-dimensional parallel coordinate plots. *Journal of Visual Languages & Computing*, Vol. 43, pp. 1–13, 2017.

[8] Z. Zhou, Z. Ye, J. Yu, and W. Chen. Cluster-aware arrangement of the parallel coordinate plots. *Journal of Visual Languages & Computing*, Vol. 46, pp. 43–52, 2018.

[9] G. Grinstein, M. Trutschl, and U. Cvek. High dimensional visualizations. In *In Proceedings of KDD Workshop on Visual Data Mining*, 2001.

[10] Melanie Herschel, Ralf Diestelkämper, and Houssem Ben Lahmar. A survey on provenance: What for? what form? what from? *The VLDB Journal*, Vol. 26, No. 6, pp. 881–906, Dec 2017.

[11] Manuela Waldner, Stefan Bruckner, and Ivan Viola. Graphical histories of information foraging. *Proc. of the 8th Nordic Conf. on Human-Computer Interaction: Fun, Fast, Foundational(NordiCHI '14)*, pp. 295–304, 2014.

[12] Peter Mindek, Stefan Bruckner, and M. Eduard Gröller. Contextual snapshots: Enriched visualization with interactive spatial annotations. *Proc. of the 29th Spring Conf. on Computer Graphics(SCCG '13)*, pp. 49–56, 2013.

[13] S. Gratzl, N. Gehlenborg, A. Lex, H. Pfister, and M. Streit. Domino: Extracting, comparing, and manipulating subsets across multiple tabular datasets. *IEEE Trans. on Visualization and Computer Graphics(TVCG)*, Vol. 20, No. 12, pp. 2023–2032, Dec 2014.

[14] P. Godfrey, J. Gryz, and P. Lasek. Interactive visualization of large data sets. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 28, No. 8, pp. 2142–2157, 2016.

[15] Mark Derthick, John Kolojejchick, and Steven F. Roth. An interactive visual query environment for exploring data. In *Proceedings of the 10th Annual ACM Symposium on User Interface Software and Technology*, UIST '97, pp. 189–198, 1997.

[16] Chris North and Ben Shneiderman. Snap-together visualization: A user interface for coordinating visualizations via relational schemata. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '00, pp. 128–135, 2000.

[17] 杉渕剛史, 田中譲. 関係データベースモデルに基づくデータベース可視化フレームワークの提案と実装. 電子情報通信学会論文誌. D, 情報・システム = The IEICE transactions on information and systems (Japanese edition), Vol. 90, No. 3, pp. 918–932, mar 2007.