SQLを拡張した情報可視化言語に対するGUI上での可換性の設計およびライフログ分析支援の有効性評価

宇津木 萌† 大久保勇輝† 川原 一輝†† 富井 尚志†††

† 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 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 大学院環境情報研究院 〒240–8501 横浜市保土ヶ谷区常盤台 79–7
E-mail: †{utsugi-moe-yx,okubo-yuki-ph}@ynu.jp ††kawahara-kazuki-cp@ynu.jp, †††tommy@ynu.ac.jp

あらまし 我々の先行研究では，PCP（Parallel Coordinates Plot）により多変量データを可視化し，その状態を SQL ライクな独自の言語表現で保存・再現可能なシステムを提案してきた．しかし, 初めに提案した $(PC)^2$DV ではデータ件数によって処理時間と描画時間が増大する問題があった．次に提案した PCPASQL + DV ではこれらの問題を解決したが，GUI 操作が不可能であった．そこで本研究では PCPASQL + DV を基盤とし，SQL を拡張した情報可視化言語と GUI の間に可換性を持つ $(PC)^2$DV V2（Parallel Co ordinates Plot Commutative Data Visualizer Version 2）を提案する．情報可視化言語として定義を行った $(PC)^2$L V2 の文法に対応した GUI を設計し，GUI 操作の状態を言語表現として保存・再現可能とした．本稿では，本システムを用いたライフログ分析例を挙げ，有効性を示す．また，先行研究とのパフォーマンス比較を行い，ビッグデータ分析が可能であることを示す．

キーワード 情報可視化，ユーザ支援，データ分析，平行座標プロット

1 はじめに

近年，センサ技術の発展や各省庁等による様々なデータの公開により，実世界の状況がデータとして取得可能になった．また，ストレージの大容量化，低価格化によって取得したデータの全てを蓄積・保存することが可能になった．これにより，人々の日常的な生活や活動を記録した「ライフログ」をデータとして容易に取得し，利用できるようになった．ライフログには個人ごと場所ごとの特徴が含まれており，そのようなデータ固有の知見を示すことは，個々の事例においても社会全体においても有用である．また，今後の社会では，様々なソースから多様なデータが取得できるようになると考えられ，日々のデータを蓄積するライフログは膨大かつ多様な多変量データとなる．このような，出処や形式，粒度，種類が異なるデータを結合して分析を行うことは重要である．以上のことから，関係データベース（RDB：Relational Database）および SQL を用いた分析が有効であるといえる．しかし，SQL による RDB の操作結果は表形式であり，その形式のまま知見を得ることは容易ではない．また，大量のデータから有用な知見を得るためには，操作を繰り返し行う必要がある．そのため，データを可視化しながら，試行錯誤を伴う分析をすることが必要であると考えられる．

そこで，我々の先行研究 [1–4] では，平行座標プロット（PCP：Parallel Coordinates Plot）[5,6] を用いたデータ可視化システムを提案してきた．PCP ではデータの属性を平行な軸に割り当て，軸を結ぶ折れ線 1 本でデータ 1 件を表す．PCP でデータ全件を可視化することにより，大まかな属性間の相関やクラスタ，外れ値を容易に把握することができる．

PCP で可視化するデータは表形式であるため，PCP 上で SQL と同等の表に対する操作が可能である．この特徴に注目し，我々は複数の属性からなるデータを PCP により可視化し，その状態を SQL ライクな言語表現により保存・再現することが可能なシステム $(PC)^2$DV （Parallel Coordinates Plot Commutative Data Visualizer）を提案してきた [1–3]. $(PC)^2$DV は GUI 上での選択・射影・結合といったリレーショナル代数演算によるデータ操作が可能である．また，分析の過程において，分析者はデータ操作や データ可視化の状態を，SQL ライクな独自の言語表現 $(PC)^2$L（Parallel Coordinates Plot Commutative Language）により 保存・再現することが可能である．

しかし，$(PC)^2$DV では件数が膨大なデータを扱う場合，データの処理時間と描画時間が長いという 2 つの問題があった．その原因として，$(PC)^2$L ではデータを取得するための SQL 部分と，取得したデータの PCP 表示方法を記述する可視化部分が混在していたことが挙げられる．$(PC)^2$DV は最初にデータを全て取得し，SQL に該当する部分をクライアントのフロントエンドで独自に処理を行っていたため，データ処理が最適化されておらず，時間がかかっていた．また，PCP の線の描画に，描画ライブラリである D3.js が用いられていたが，D3.js では CPU で線の描画を行うため，GPU を活用できていなかった．そのため，PCP の描画時間が長くなっていた．そこで先行研究 [4] では，これらの問題を解決した新たなデータ可視化ツールである PCPASQL + DV（Parallel Coordinates Plot Augmented SQL + Data Visualizer）を提案した．PCPASQL + DV では 2 つの改善を行った．1 つ目として，新たに情報可視化言語である PCPASQL（Parallel Coordinates Plot Augmented SQL）を定義した．PCPASQL では，SQL 部分と可視化部分を分離

し，SQL 部分は DBMS に処理させることで処理速度を向上させた．2 つ目の改善として，実装に OpenGL を使用したことがあげられる．それにより， PCPASQL + DV では， OpenGL による GPU アクセラレーションを用いることで描画速度を向上させた．

PCPASQL + DV はこれらの改善に注力したため，(PC)^{2}DV では可能であった GUI 上での操作が実装されておらず，PCPASQL のクエリ文入力でのみ操作が可能であった．そのため，表示された PCP を用いた直観的な操作ができず，試行錯誤を伴う分析には適していないと考えられる．そこで，本研究では PCPASQL + DV を基盤とし，GUI による PCP の操作を追加した (PC)^{2}DV V2（Parallel Coordinates Plot Commutative Data Visualizer Version 2）を提案する．(PC)^{2}DV V2 は，GUI 上で直感的な操作を可能とし，操作結果を独自の情報可視化言語である (PC)^{2}L V2（Parallel Coordinates Plot Commutative Language Version 2）で保存・再現ができるデータ分析支援ツールである．GUI の操作手順はリレーショナル代数を参考に設計し，基本演算に対応したものとする．これにより，GUI による操作で RDB に対する操作を広範でカバーすることが可能になり，SQL を拡張した言語である (PC)^{2}L V2 で操作過程を表現できるようになる．また，ライフログデータ分析の例として，建物の需要電力データの分析を行う．これにより， 試行錯誤を伴うライフログデータ分析における (PC)^{2}DV V2 の有用性を示す．

2 関 連 研 究

2.1 データ可視化とデータ分析

PCP は 1985 年，Inselberg によって初めて概念が定義された [5]．それ以降，PCP を用いたデータ分析の研究が盛んに行われている．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 [6]：

（1） PCP の（属性）軸レイアウト
（2） PCP の Clutter 軽減方法
（3） PCP の実応用例の提示
（4） PCP と他のデータ解析手法との比較

しかし，PCP の見せ方に関しての議論がほとんどであり，PCP の操作過程に着目した議論はされていない．また，操作の過程でデータ自体にリレーショナル代数のような演算を加えながら操作するものは議論されていない．

複数の属性からなるデータを可視化するその他の一般的な手法として，複数の散布図を表示する散布図行列が挙げられる [7]．散布図行列は，2 つの属性間の相関を直感的に把握できるが，散布図数が属性数の 2 乗に比例して増加する．そのため，分析過程でデータに対し結合の操作を行うことには不向きであるといえる．また，Bouali らは，対話型遺伝的アルゴリズムにより可視化手法の推薦を行い，データや利用者の要求に応じてより適切な可視化手法の選択を支援するシステムを提案した [8]．我々が提案してきた (PC)^{2}DV，PCPASQL + DV と，本稿で提案する (PC)^{2}DV V2 は関係代数演算における選択・射影・結合が表現可能な可視化システムであるため，データ一件を 1 本の折れ線で表し，詳細に参照・分析可能である PCP が適切である．データを可視化し，分析を行う研究（Visual Analytics）が盛んに行われている．Cui [9] による分類では，PCP を用いたデータ分析および本研究は多次元データをアルゴリズムに基づき変形し二次元空間で可視化する”Multi-Dimensional-Transformation-2D”かつ，データ操作によりデータや可視化空間を探索する”Exploratory-Oriented”に分類される．

Visual Analytics の中で，我々の手法と同様にインタラクティブな操作と PCP による可視化を組み合わせた可視化・分析手法の提案がされている．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 [10]．Zhou らは，エントロピーの概念を導入することで，PCP の属性軸の整列順序をクラスタに基づいて決定する手法を提案した [11]．Bok らは，任意の属性値を基準としたデータの分布を表すヒストグラム（PHP：Parallel Histogram Plot）を PCP 上に表示し，PCP の軸上のデータの分布や属性間の相関の把握を支援する手法を提案した [12]．Gruendl らは，時間を新たな次元と考え，PCP の二軸間の奥行き方向に時間軸を導入し，PCP と時系列プロットを統合した，時間依存のデータを可視化・分析する手法を提案した [13]．Cibulski らは，相互関係のあるデータセットの PCP を結合した複合平行座標により，複数のコンポーネントを同時に分析する手法を提案した [14]．これらの研究と比較して我々は，PCP 上でのデータ操作が関係代数演算と同等であり，可視化の状態と言語が可換である点に着目している．

2.2 データ操作過程管理（Data Provenance）

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 [15]．Herschel らは文献 [15] 内で，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操作の意図を保存することは，複雑なデータ処理を支援するために重要なことであると述べている．さらに，分析結果データの操作過程や操作の意図を示すことは，SQL のような関係代数演算をサポートする問合せ言語で記述することが有効であるとも述べている．この点において，(PC)^{2}L V2 を用いて (PC)^{2}DV V2 のデータの操作過程の状態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が提案されている．Waldner らは，PC のアプリケーションの閲覧履歴や操作履歴を保存し，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 [16]．Mindek らは，画像データと，分析過程で利用する他のソースのデータを同時に表示し，分析者の文脈を含蓄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 [17]．Psallidas らは，インタラクティブ性を持つ可視化ツールの操作過程を宣言的操作クエリに変換することによる有用性を述べた [18]．Gratzl らは，PCP やヒートマップ，散布図行列など様々な可視化手法を組み合わせて複数のソースか

ら得られたデータとその解析過程を可視化し，データ解析の支援を行った [19].

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解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過程の任意の状態を保存し，問合せ言語として一般的な SQL を熟知す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，言語を用いて状態を保存することにより，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，Holger らは分析プロセスの可視化状態と操作を，検索可能なグラフ構造として保存・再現する分析支援システムを提案した [20]．彼らの手法は，分析における過去の状態を言語情報により保存・再現可能である，という点が我々と共通する．しかし，彼らは「グラフ構造により分析の履歴を示し」，操作を含めた分析過程を「検索可能な形ですべて保存する」一方で，我々の手法は「可視化状態と言語が可換である」という点を重視し，分析過程において GUI 上で構成された可視化結果の状態をクエリ言語の形式でことにより支援を行う．この点において，我々の研究とは立場が異なる．

3 $(PC)^2$DV と PCPASQL + DV における問題

我々は先行研究 [1–3] において PCP によるデータ可視化システムである $(PC)^2$DV を提案してきた．さらに，その後の先行研究 [4] では $(PC)^2$DV の問題点を改善した PCPASQL + DV を提案した．これらの先行研究と本稿で提案する $(PC)^2$DV V2 をまとめた表を表 1 に示す．本章では $(PC)^2$DV と PCPASQL + DV の概要と問題について説明する.

3.1 $(PC)^2$DV の概要

$(PC)^2$DV では，PCP に対し GUI 上でインタラクションを行い，SQL ライクな独自の言語 $(PC)^2$L によりデータ操作を行った状態を保存・再現することが可能である．$(PC)^2$DV は Web ブラウザを通して多くの端末から利用できるように構築した．このシステムの操作の流れを以下に示す.

（1） 任意のデータソースへ接続し，データを読み込む.

（2） リレーションを PCP により可視化する.

（3） PCP に対して GUI 上でインタラクション（データ操作）を行う．この際，データ操作結果はリアルタイムに反映される.

（4） データ分析者が任意に,(2) ,(3) の分析過程の状態を $(PC)^2$L で保存する.

（5） (2) から (4) を繰り返す．その際，過去の状態に戻りたい場合は該当する $(PC)^2$L を入力し，その状態を再現する.

（6） データ分析者が所望の可視化結果を獲得する.

$(PC)^2$DV には 2 つの問題が存在した．それは，データ処理時間が長いこととデータ描画時間が長いことである.

1 つ目の問題として，分析対象データの処理に時間がかかってしまうことがあげられる．$(PC)^2$DV では GUI 上での操作や，$(PC)^2$L の入力結果のデータを作成する処理をフロントエンドで独自に行っていた．$(PC)^2$DV の分析対象となる多変量データは JavaScript の連想配列となり，インデックスを利用したデータ構造にはなっていない．また，クエリの最適化処理も行われていない．そのため，データ件数が増えるにつれて，データ操作処理の時間は線形時間で増大してしまう.

2 つ目の問題は，データ処理を行った後の．PCP が描画されるまでの時間が長いことである．$(PC)^2$DV ではデータ描画の API として D3.js が使われていた．D3.js は主に SVG を用いて描画を行う．また，DOM ツリーを介してブラウザ上に要素をレンダリングするため，処理は主に CPU に依存する．その結果，描画する線や要素の数が増えるとデータ描画時間が長くなる問題がある．これは，データ件数が膨大になるライフログデータの分析において支障をきたすと考えられる.

3.2 PCPASQL + DV の概要

PCPASQL + DV は，$(PC)^2$DV のデータ処理時間とデータ描画時間の問題を改善した可視化システムである．このシステムは，独自の情報可視化言語である PCPASQL を用いて，データを PCP として可視化し，分析を行うことができる．以下のような手順で分析を行う.

（1） データ分析者が事前に PCPASQL を指定のファイルに記述する.

（2） 指定された PCPASQL に従って，PCPASQL+DV が PCP を描画する.

PCPASQL + DV では，$(PC)^2$DV で問題となっていたデータ処理時間とデータ描画時間に対して 2 つの改善を行った．1 つ目として情報可視化言語である PCPASQL を提案し，SQL 部分（データ処理）と可視化部分（描画設定）を明確に分けた．SQL 部分と可視化部分に分けることで，SQL 部分の処理を全て DBMS に任せることが可能になり，以下の機能を活用できるようになった．これにより，データ処理速度を向上させることが可能となった.

- インデックス構造の活用
- 実行プランの最適化

2 つ目の改善として実装に OpenGL を用いた．OpenGL とはグラフィックを描画するための API であり，C,C++などの言語で利用可能である．OpenGL による GPU アクセラレーションの結果として PCP の線の描画速度を向上させた.

PCPASQL + DV では，ビッグデータに利用可能なデータ分析システムを実装することを優先した．そのため，PCP と情報可視化言語 PCPASQL の間に可換性を持たせることを諦めた．また，GUI 上での操作機能は実装されておらず， PCPASQL のクエリ文を入力することでのみデータ操作を実行可能であった．つまり，$(PC)^2$DV の特徴である，リアルタイムにデータ操作が反映される GUI と，その状態と等価である論理的な言語表現を相互に活用した分析が不可能であった.

表 1　先行研究の比較

可視化システム	可視化言語	可換性	GUI 上での操作	使用ライブラリ	文献
$(PC)^2$DV	$(PC)^2$L	○	○	D3.js	[1–3]
PCPASQL + DV	PCPASQL	×	×	OpenGL	[4]
$(PC)^2$DV V2	$(PC)^2$L V2	○	○	OpenGL	本稿

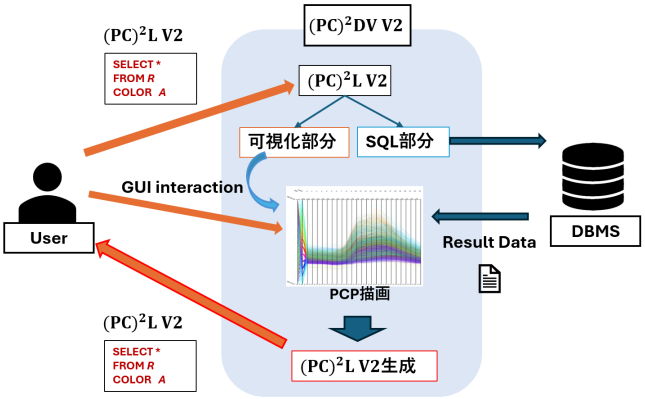

図 1　$(PC)^2$DV V2 の概要

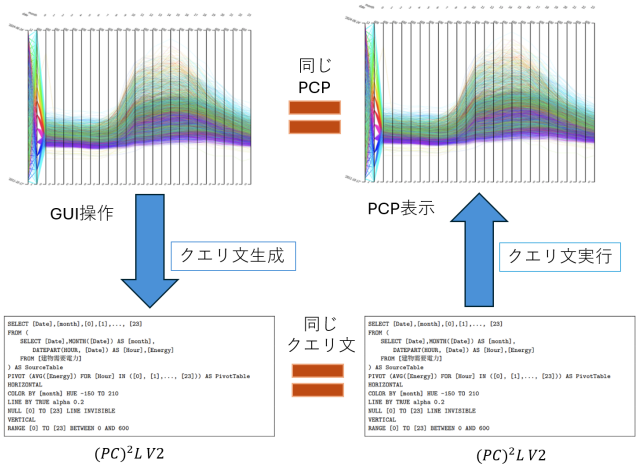

図 2　GUI と $(PC)^2$L V2 の可換性の概要

4　$(PC)^2$DV V2 の概要と設計

本章では，新たに提案するデータ可視化システムである $(PC)^2$DV V2 の概要と設計について述べる．

4.1　$(PC)^2$DV V2 の概要

$(PC)^2$DV V2 の概要を図 1 に示す．$(PC)^2$DV V2 では，先行研究 [4] で提案した PCPASQL + DV の機能に追加し，PCP に対し GUI 上でインタラクションを行いつつ，その操作結果を言語によって保存・再現することを可能にする．本システムでは，新たに情報可視化言語 $(PC)^2$L V2 を定義し，GUI と $(PC)^2$L V2 が可換性をもつように設計する． 想定する可換性の概要を図 2 に示す．これにより，ライフログのような膨大な件数を持つデータの分析に耐えうる，直感的な操作で試行錯誤が可能なデータ可視化システムを実現する．$(PC)^2$DV V2 では，$(PC)^2$DV と同様に，以下のようなデータ操作手順により分析を行う分析者に対して支援を行うことを想定する．

（1） 任意のデータソースへ接続し，データを読み込む．

（2） リレーションを PCP により可視化する．

（3） PCP に対して GUI 上でインタラクション（データ操作）を行う．

（4） データ分析者が任意に，(2)，(3) の分析過程の状態を $(PC)^2$L V2 で保存する．

（5） (2) から (4) を繰り返す．その際，過去の状態に戻りたい場合は該当する $(PC)^2$L V2 を入力し，その状態を再現する．

（6） データ分析者が所望の可視化結果を獲得する．

4.2　PCP に対する SQL 操作

PCP ではリレーションの可視化を行うため，SQL のリレーショナル代数演算の操作が可能である．本研究では，リレーショナル代数演算である選択演算，射影演算，直積演算の 3 つを PCP 上での SQL 操作として考える．リレーショナル代数演算については，文献 [21] を参考にした．本節では，これらの演算と PCP 上での操作の関係を述べる．ここで用いる記号の定義を表 2 に示す．

選択演算　リレーションにおける選択演算は，属性値に基づいてタプルを抽出する操作である．これは，PCP 上に表示された折れ線のうち，条件を満たすものを選択的に表示する操作に対応する．SQL では WHERE 句に相当し，属性 A_i に対して行われる操作は「WHERE $p(A_i)$」の形式で記述される．ここで p は 「$A_i \theta v$」，「A_i BETWEEN v_1 AND v_2」，「A_i LIKE s」などの条件式である．

射影演算　射影演算は，リレーション $R(A_1, A_2, \cdots, A_n)$ から，$R[A_{i1}, A_{i2}, \cdots, A_{ik}]$ のように属性を切り出す操作である．PCP 上では軸を切り出す操作に対応する．SQL では SELECT 句による属性の列挙に相当し，列挙された順序に従って，PCP の軸が左から右へ配置されるものとする．

直積演算　直積演算は，2 つのリレーション $R(A_1, A_2, \cdots, A_n)$ と $S(B_1, B_2, \cdots, B_m)$ から，すべてのタプルの組合せを生成し，新たなリレーション $R \times S(A_1, A_2, \cdots, A_n, B_1, B_2, \cdots, B_m)$

表 2　記号の定義

記号	概要
A_i	属性
$R(A_1, A_2, \cdots, A_n)$	リレーション
$R[A_i \theta A_j]$	R の A_i と A_j 上の θ-選択演算
$R[A_{i_1}, A_{i_2}, \cdots, A_{i_k}]$ $(1 \leqq i_1 < i_2 < \cdots < i_k \leqq n)$	R の $A_{i_1}, A_{i_2}, \cdots, A_{i_k}$ 上の射影演算

を構成する操作である．PCP 上では，2 つのリレーションに対応する折れ線集合を組み合わせ，すべての組合せとして表示する操作に対応する．SQL では FROM 句における複数テーブルの列挙に相当し，その後に WHERE 句で条件を付与することにより結合演算として実現される．

4.3 PCP 可視化の独自操作

本節では，PCP 可視化に対する独自操作について述べる．PCP に対する操作は水平方向の折れ線に対するものと，垂直方向の軸に対するものに分けられる．

4.3.1 折れ線に対する操作

折れ線に対する操作は.「線の色付け」,「線の太さや種類の変更」,「NULL 値の扱いの変更」の 3 つに分けられる．

「線の色付け」は，任意の属性 A の値を基準として折れ線の色分けを行う操作である.「線の太さや種類の変更」では，色分けを行う場合と同様に，任意の属性 A の値を基準として折れ線の太さ，透明度，線種などを変更する.「NULL 値の扱いの変更」は，NULL 値を含むデータを PCP の折れ線としてどのように描画するかを決定する操作である．

4.3.2 軸に対する操作

軸に対する操作は.「軸の色付け」,「軸の太さや種類の変更」,「軸の範囲変更」の 3 つに分けられる．

「軸の色付け」は，折れ線に対する操作と同様に，任意の属性 A の値を基準として軸の色を変更する操作である.「軸の太さや種類の変更」も，折れ線に対する操作と同様に，任意の属性 A の値を基準として軸の太さ，透明度，線種などを変更する.「軸の範囲変更」は，軸の最大値および最小値を指定して表示範囲を変更する操作である．軸の範囲を狭めることで，折れ線の傾向をより詳細に可視化することが可能となる．

4.4 情報可視化言語 $(PC)^2L$ V2

本節では，$(PC)^2$DV V2 で使用する情報可視化言語である $(PC)^2$L V2 について述べる．PCP のインタラクションの過程を保存・再現するためには，PCP の GUI 上での各操作と対応する言語の定義が必要である．そこで，本稿では情報可視化言語 $(PC)^2$L V2 を提案する．$(PC)^2$L V2 は，PCP におけるリレーショナル代数表現と対応し PCP における操作結果を言語化して保存する機能と，言語をもとに PCP を描画する機能を有する．また， PCPASQL + DV と同様に SQL 部分と可視化部分が分かれており，SQL 部分を分離して DBMS に送信することでデータ処理の高速化が可能である．

本言語の SQL 部分では，リレーショナル代数の基本演算が表現可能であるものとする．基本演算として，4.2 節で述べたように，選択演算，射影演算，結合演算をそれぞれ SQL 部分の WHERE 句，SELECT 句，FROM 句として表現し，PCP におけるインタラクションと対応するように定義する．以下では，これら基本演算と PCP の関係について述べる．

- **WHERE 句：**選択演算．軸に対して条件や範囲選択することで，表示する PCP の線（タプル）を絞り込む．
- **SELECT 句：**射影演算．テーブルのカラムから PCP で表示する軸を指定し，任意の順番に並べる．
- **FROM 句：**直積演算．複数のテーブルを読み込んだ場合，それぞれが持つ線（タプル）をつないで軸の追加を行う．

これらに加え，可視化部分として，PCP の描画方法を記述する句である HORIZONTAL 句と VERTICAL 句も利用可能とする．HORIZONTAL 句と VERTICAL 句の仕様に関して，以下に示す．

- **HORIZONTAL 句**
  - **COLOR 句：**「COLOR A_i WITH」で軸を指定し，軸の値を基準にして線を色付ける．色付けの方法はグラデーションと条件式を記述する 2 つがある．グラデーションでは,「HUE」（色相）,「SATUATION」（彩度）,「VALUE」（明度）,「ALPHA」（透明度）のどれかを選択し，最小値・最大値を記述する．条件式では，その後ろに「赤」,「緑」,「青」を記述することも可能である．
  - **FIX LINE 句：**線の太さや透明度，線種の変更を記述する.「FIX LINE IN [条件] TO」とすることで，変更する対象となる PCP の線を指定する．線の太さを変更する場合は「WIDTH [太さ]」，透明度は「ALPHA [透明度]」，線種を点線にする場合は「DASHED」で記述する．
  - **FIX NULL 句：**リレーションの NULL を PCP の線でどのように描画するか記述する.「FIX NULL IN A_i TO」で対象の軸の指定を行う．描画方法は「TOP」,「BOTTOM」,「INTERPOLATE」,「VALUE INVISIBLE」,「LINE INVISIBLE」の 5 つである．
- **VERTICAL 句**
  - **COLOR 句：**「COLOR A_i」で色付けを行う軸を指定する．
  - **FIX LINE 句：**軸の太さや透明度，線種の変更を記述する.「FIX LINE IN A_i TO」とすることで，変更する対象となる軸を指定する．太さを変更する場合は「WIDTH [太さ]」，透明度は「ALPHA [透明度]」，線種を点線にする場合は「DASHED」，非表示にする場合は「INVISIBLE」で記述する．
  - **SCALE 句：**軸の範囲を記述する.「SCALE [軸] BETWEEN [最小値] AND [最大値]」とすることで，軸の範囲を最小値から最大値で描画することが可能である．対象となる軸は「A_i」と 1 本のみではなく,「A_{i_1} TO A_{i_1} 」のように 2 本記述することで，この 2 本の間に存在する軸も含めて範囲を指定することができる．

4.5 PIVOT 操作を用いた時系列データ分析

PIVOT とは SQL において，リレーション名の行を削減し列に変換する操作である．この操作により，データを束ね件数を減らしつつ属性数を増やすことでデータ一件の単位を変えることが可能である．たとえば，データ 1 件が 1 時間を表すリレーションを，データ 1 件で 1 日を表す周期データへ変換することができる．

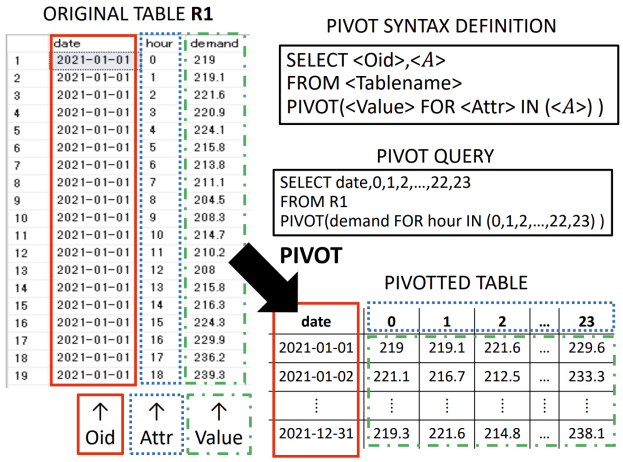

図 3 PIVOT 操作の概要

PIVOT 操作を実行するにあたり，必要な要素として< Oid >, < Attr >, < Value >がある．< Oid >とは，PIVOT 後のテーブルの主キーとなる属性であり，< Oid >の値ごとにデータが束ねられる．< Attr >とは，PIVOT 後のテーブルのカラム名となる値を持つ属性である．< Value >とは，PIVOT 後のテーブルの各カラムの値となる値を持つ属性である．

図 3 では属性 [date]，[hour]，[demand] からなるテーブル R1 に対して PIVOT の操作を行う．< Oid >となる属性は [date] であり，PIVOT 後のテーブルはデータが日ごとに束ねられ，レコード 1 件が日ごとのデータとなる．< Attr >となる属性は [hour] であり，この値が PIVOT 後のテーブルのカラムとして並ぶ．< Value >となる属性は [demand] である．このようにして [date] ごとに [hour] ごとの [demand] の値が PIVOT のテーブルの値として並ぶ．

我々が分析対象とするライフログは，時系列データの形式として蓄積される．そのため，PIVOT 操作を行うことで，時系列データを周期データに変換し，PCP で折れ線グラフのような分析が可能になる．

5 (PC)^{2}DV V2 を用いたライフログ分析

5.1 分析の目的

本章では，4 章で述べた (PC)^{2}DV V2 を用いて建物の需要電力データを分析する．需要電力データは，1 時間ごとに取得されている．このデータを PIVOT 操作により，1 日ごとの周期データに変換し PCP で日々の電力波形を可視化する．これにより，需要電力の季節ごとの違いや，時刻による傾向を可視化する．次に，曜日のデータと結合し，休日と平日の需要電力の違いを可視化する．また，(PC)^{2}L V2 により保存したデータ操作結果を利用して，複数の建物の需要電力の傾向を比較する．以上について PCP を操作して説明することで，ライフログデータ分析における (PC)^{2}DV V2 の有用性を示す．

5.2 使用データと分析方法

本稿では，分析対象の建物の需要電力データとして，横浜国立大学の総合研究棟（建物 A）の 1 時間ごとの需要電力の実データ [1]を使用する．データ取得期間は 2019 年 1 月 1 日から 2019 年 12 月 31 日まであり，データ件数は 24 時間 × 365 日で 8760 件である．このリレーション「建物需要電力_建物 A」の属性について表 3 に示す．なお，データは Microsoft SQL Server 上のデータベースに格納されている．このデータベースに ODBC 接続し，データを取得する．

表 3 リレーション「建物需要電力_建物 A」の属性

属性名	説明
month	月 （mm）
day	日付（dd）
hour	時間（0〜23）
demand	1 時間ごとの建物の需要電力量（kWh/h）

表 4 PIVOT 操作後のリレーション「建物需要電力_建物 A」の属性

属性名	説明
month	月 （mm）
day	日付（dd）
0〜23	1 時間ごとの建物の需要電力量（kWh/h）

表 5 リレーション「曜日」の属性

属性名	説明
month	月 （mm）
day	日付（dd）
weekday	曜日

5.1 節で述べたように，取得されたデータは時系列データになっており，そのまま PCP で可視化しても知見を得ることは難しい．そこで，日付（[date]）を主キーとして，時間（[hour]）がカラム名となるように PIVOT 操作による変換を行う．変換を行い周期データとなったリレーション「建物需要電力_建物 A」を表 4 に示す．

このリレーション [建物需要電力_建物 A] について，(PC)^{2}DV V2 を用いて以下のプロセスで可視化・分析を行う．結合するリレーション「曜日」を表 5 に示す．

（1） データソースに接続してデータを取得し，PCP で可視化する．

（2） 可視化されたデータに対して，PIVOT の操作を行い，日ごとの周期データに変換する．

（3） 変換されたデータを PCP で可視化する．

（4） 同一の尺度で値を比較するため，軸の範囲を統一する．

（5） 季節ごとの傾向を見るため，月を基準に折れ線を色付けする．

（6） リレーション「曜日」と結合する．

（7） 結合した曜日を基準に折れ線を色付けする．

5.3 需要電力データの分析

まず，1 つの建物の需要電力データ「建物需要電力_建物 A」について (PC)^{2}DV V2 上での操作・分析を行う．5.2 節で述

1：横浜国立大学施設部，http://shisetsu.ynu.ac.jp/gakugai/shisetsu/（学内限定アクセス）

べたように，リレーション「建物需要電力_建物 A」はそのまま可視化しても周期的な情報を読み取るのは難しい．そこで，PIVOT 操作による変換を行い，1 日ごとの周期データとして可視化する．変換後の PCP とその状態を表す $(PC)^2L$ V2 によるクエリ文を図 4 に示す．図 4 では，周期データとして可視化できているが，[0] から [23] までの各軸の範囲にばらつきがあるため，軸の範囲を指定して揃える操作を行う．軸の範囲を 0kWh/h から 500kWh/h に揃えた PCP を図 5 に示す．これにより，1 日の電力推移を波形として把握することができる．この図から，この建物の需要電力は人々の活動に合わせて 7 時ごろから増加し，昼過ぎから夕方の間に大きくなり，夜になるにつれて減少するという傾向があることがわかる．さらに，折れ線が上下方向に分布していることから，日によって需要電力に違いがあることがわかる．

ここで，軸 [month] を基準にして折れ線に色を付ける操作を行う．操作後の PCP は図 6 のようになり，季節ごとの大まかな違いを把握することができる．この図から，夏（緑）や冬（青）の昼間の需要電力は大きく増加することがわかる．また，春（赤）の需要電力は他の季節と比べて低いことがわかる．さらに，春（赤）の折れ線は，昼間に大きく増加するクラスタと，増加が比較的小さいクラスタに分かれていることが確認できる．

この 2 つのクラスタに分かれる原因として，休日と平日の違いが関係していると考え，その違いを明確化するために，リレーション「曜日」の結合を行う．結合を行うために，リレーション「建物需要電力_建物 A」に追加して，リレーション「曜日」のデータを取得し可視化する．ただし，単に可視化を行った状態では直積演算をとっていることになり，すべてのタプルの組合せで繋がっている状態である．そのため，選択演算で日付が一致している折れ線のみを抽出する操作を行う．この操作により，曜日ごとの需要電力の傾向を可視化することができる．土曜日と日曜日を青色，月曜日から金曜日を赤色とした PCP を図 7 に示す．この図を見ると，土日を表す青色の線が下部に集中し，平日を表す赤色が上に広がっていることが確認できる．よって，休日は需要電力が低くなり，反対に平日は高くなることがわかり，春の需要電力が上下 2 つのクラスタに分かれている原因が，休日と平日の違いであると確認できる．このように，データ操作を行いながら PCP で描画することで，季節ごとの差異や休日と平日の違いを可視化することが可能である．

5.4 $(PC)^2L$ V2 による操作結果の利用例

$(PC)^2DV$ V2 上での PCP の操作後の状態は言語 $(PC)^2L$ V2 のクエリ文で保存可能である．保存したクエリ文は，同じ構造の異なるリレーションに対して使用することができる．5.3 節では，総合研究棟（建物 A）の需要電力について分析を行ったが，この操作結果を保存し，別の建物の需要電力のデータに適用することが可能である．

本節では，図 6 の状態を保存した $(PC)^2L$ V2 のクエリ文を利用して，別の建物の需要電力データを可視化し，比較を行う．比較する建物は，横浜国立大学の環境情報 1 号棟（建物 B）と，経済・経営系（建物 C）である．これらの建物の需要電力データのリレーションは，「建物需要電力_建物 A」と同じ構造をしている．

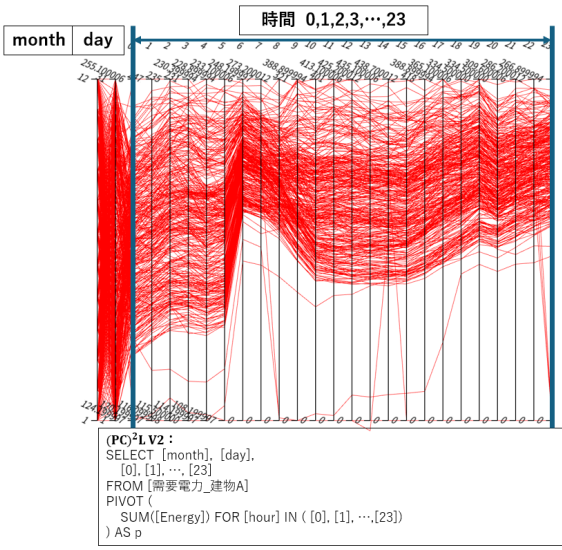

図 4 「建物需要電力_建物 A」を PIVOT で変換したの PCP

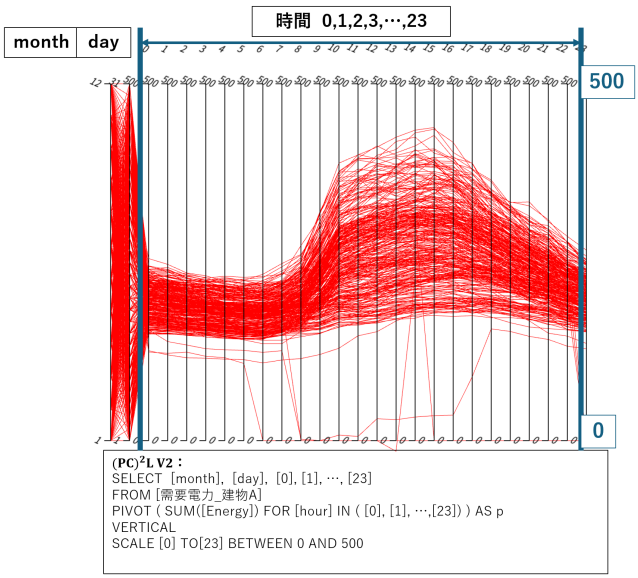

図 5 「建物需要電力_建物 A」の軸を揃えた PCP

まず，保存したクエリ文にある FROM 句のリレーション名を，環境情報 1 号棟（建物 B）のデータである「建物需要電力_建物 B」に変更する．変更したクエリ文を図 8 に示す．このクエリ文を $(PC)^2DV$ V2 で実行することで，図 9 の PCP が描画される．このように，$(PC)^2L$ V2 により 5.3 節で行った操作を再び行うことなく，クエリ文を実行するだけで，操作の再現が可能である．この図を図 6 と比較することで，建物ごとの需要電力の違いを把握できる．さらに，経済・経営系（建物 C）のリレーション「建物需要電力_建物 C」も PCP で可視化する．「建物需要電力_建物 B」の分析と同様に，保存したクエリ文のリレーション名を「建物需要電力_建物 C」に変更し，$(PC)^2DV$ V2 で実行する．これにより描画された PCP を図 10 に示す．

ここで，建物 A の図 6 の PCP と，建物 B の図 9 を比較す

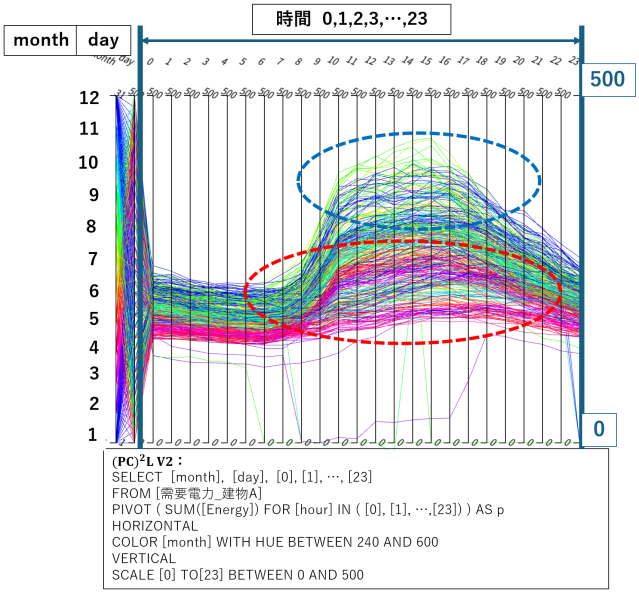

図 6 「建物需要電力_建物 A」の月を基準に色分けした PCP

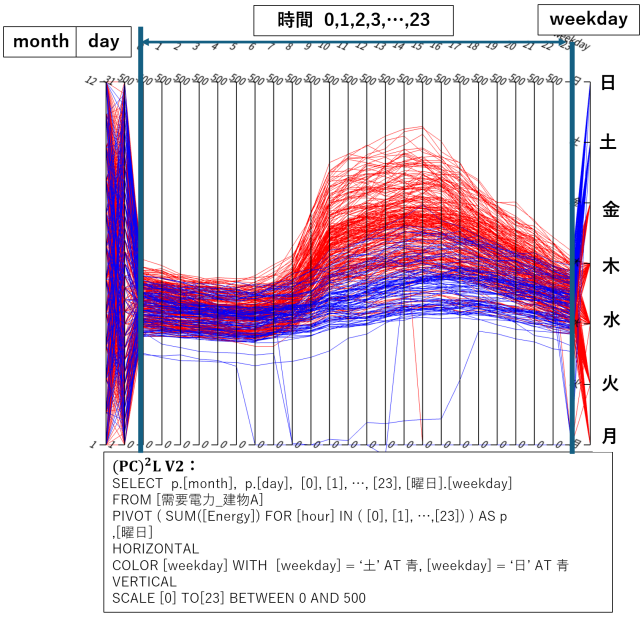

図 7 「建物需要電力_建物 A」と「曜日」を結合した PCP

```
SELECT [month], [day], [0], [1], ⋯, [23]
FROM [需要電力_建物B]
PIVOT ( SUM([Energy]) FOR [hour] IN ( [0], [1], ⋯,[23]) ) AS p
HORIZONTAL
COLOR [month] WITH HUE BETWEEN 240 AND 600
VERTICAL
SCALE [0] TO[23] BETWEEN 0 AND 500
```

図 8 リレーション名を変更した $(PC)^2$L V2 クエリ文

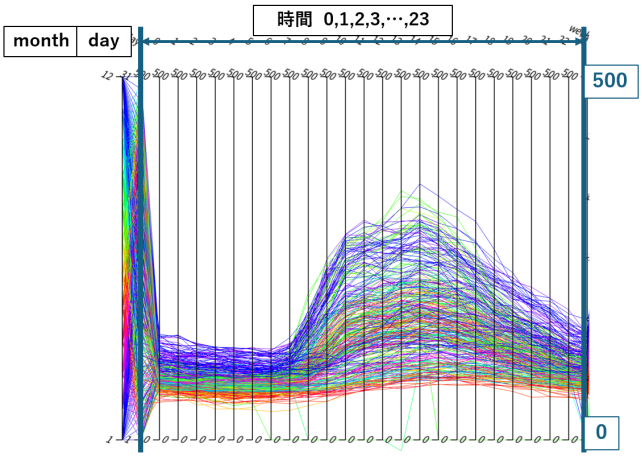

図 9 「建物需要電力_建物 B」の PCP

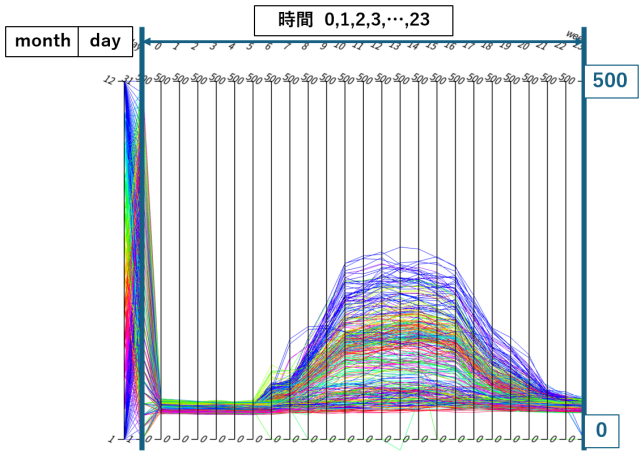

図 10 「建物需要電力_建物 C」の PCP

ると，おおまかな傾向は同じであることがわかる．しかし，夜間の需要電力に差がある．この理由として，建物 A は夜間も稼働し続けているサーバや実験機器などの数が多いことが考えられる．また，建物 A の図 6 の PCP と，建物 C の図 10 を見比べると，傾向が異なっているのがわかる．建物 C の夜間における需要電力は日ごとの差異が小さく，ほぼ一定であることが確認できる．また，夕方から夜間にかけて需要電力が急激に減少している様子が把握できる．これは，建物 C では夜間に稼働している機器がほとんど存在せず，夜遅くまで滞在する人が少ないことが要因であると考えられる．本稿では，$(PC)^2$DV V2 を用いて 3 つの建物需要電力データの分析を行ったが，リレーションの構造が同一であれば，他の建物データに対しても同様の分析が可能である．

6 パフォーマンス評価

本章では，$(PC)^2$DV V2 と $(PC)^2$DV,PCPASQL + DV のパフォーマンスの比較を行い，$(PC)^2$DV V2 によるビッグデータ分析が可能であることを示す．

6.1 $(PC)^2$DV V2 におけるパフォーマンスの定義

$(PC)^2$DV V2 のパフォーマンスには，SQL を実行して DBMS からデータを取得するクエリ処理時間と，取得したデータを PCP として描画するデータ描画時間の 2 つがある．ここで．ユーザの操作終了から PCP で描画するデータセットを作成するまでをクエリ処理時間，画像処理を開始してから，画像処理が終了するまでの時間をデータ描画時間と定義した．クエリ処理時間とデータ描画時間を合わせたものが合計時間となる．

表 6　クライアント PC の実験環境

項目	説明
OS	Windows 11 Education
CPU	Intel(R) Core(TM) i5-13400
メモリ	32GB
GPU	Intel UHD Graphics 730

表 7　サーバ PC の実験環境

項目	説明
OS	Windows Server 2022
DBMS	Microsoft SQLServer 2022 developer
CPU	Intel(R) Core(TM) i7-13700 2.10GHz
メモリ	128GB
ディスク	WD BLACK SN850 HS 2TB × 3
ネットワーク転送速度	2.5Gbps

表 8　実験用テーブルの属性

属性	インデックス	データ型	値の範囲
id_index	あり（クラスター化）	int	1–10,000,000
int_data_index	あり（非クラスター化）	int	1–30
float_data	なし	float	1–10

```
SELECT [id_index],[int_data_index],[float_data]
FROM [TableName]
WHERE [id_index] <= N
```

図 11　実験で用いたクエリ

6.2 評価概要

本実験ではクライアント PC で (PC)^{2}DV V2, (PC)^{2}DV. PCPASQL + DV を実行して，サーバ PC から実験用テーブルのデータを取得して実験を行う．クライアント PC の環境を表 6，サーバ PC の環境を表 7 に示す

サーバ PC の Microsoft SQL Server 2022 に，実験用テーブルを作成した．この実験用テーブルのデータ件数は 10,000,000 件であり，属性数は 3 である．各属性名は，[id_index],[int_data_index],[float_data] である．各属性の詳細を表 8 に示す．[id_index] は int 型で，1 から 10,000,000 までの値を順番に格納している．[int_data_index] は int 型で，1 から 30 までのランダムな整数値を格納している．[float_data] は float 型で，1 以上 10 未満のランダムな浮動小数点を格納している．[id_index] にクラスター化インデックス，[int_data_index] に非クラスター化インデックスを設定した．このテーブルから，データ件数 N（100 | 200 | 500 | 1,000 | 2,000 | 5,000 | 10,000 | 20,000 | 50,000 | 100,000 | 200,000 | 500,000 | 1,000,000 | 2,000,000 | 5,000,000 | 10,000,000）を取得する．

実験に用いるクエリを図 11 に示す．このクエリでは，WHERE 句によって [id_index] の値を参照してデータ数を N 件に絞り込み，結果データ数を N 件とする．これを，(PC)^{2}DV V2, (PC)^{2}DV, PCPASQL + DV で実行し，クエリ処理時間とデータ描画時間の計測を行った．

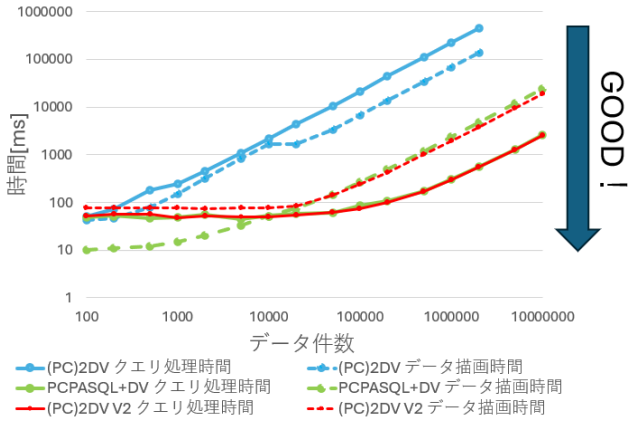

図 12　クエリ処理時間とデータ描画時間の計測結果

6.3 結果と考察

提案システムの (PC)^{2}DV V2 と先行研究である (PC)^{2}DV, PCPASQL + DV における，取得するデータ数に対しての，クエリ処理時間とデータ描画時間を測定した結果を両対数グラフとして図 12 に示す．赤の線が提案システムである (PC)^{2}DV V2 を表しており，青の線が (PC)^{2}DV, 黄緑の線が PCPASQL + DV を表している．また，実線がクエリ処理時間を，点線がデータ描画時間を表している．この図より，(PC)^{2}DV V2 はクエリ処理時間，データ描画時間ともに，速度の改善を行った PCPASQL + DV とほぼ同等となっていることが分かる．これにより，PCPASQL + DV に GUI 操作を追加した (PC)^{2}DV V2 でも，クエリ処理時間，データ描画時間が増えることなく，ビッグデータ分析に有効であることが検証された．なお，今回使用したクエリは図 11 に示すような比較的単純なものであったが，より複雑なクエリを使用する場合には，クエリ処理時間が増加する可能性がある．

7 まとめと今後の課題

我々は，複数の属性からなるデータを PCP により可視化し，SQL を拡張した言語で操作可能な可視化システムを提案してきた．しかし，先行研究である (PC)^{2}DV と PCPASQL + DV にはライフログデータ分析を行う上で，操作後の処理時間が長いことや，GUI による直感的な操作に対応していないといった問題があった．そこで，本研究ではこれらの問題を解決する (PC)^{2}DV V2 を提案した．本システムによる，リレーショナル代数をもとに設計した GUI 操作と，操作結果を保存・再現する情報可視化言語 (PC)^{2}L V2 を用いて，建物の需要電力データの分析を行い，有効性を示した．また，先行研究の実装とのクエリ処理時間とデータ描画時間の定量的な比較を行った．これにより，(PC)^{2}DV V2 によるビッグデータ分析が可能であることを示した．

今後の課題として，散布図や積み上げ棒グラフのような PCP 以外のグラフ描画機能を (PC)^{2}DV V2 に実装し，マルチビューによるデータ分析を可能とすることがあげられる．また，今回の分析で扱った電力データとは全く異なる種類のデータを対象

とした分析により，有用な知見を示していく．

謝　　辞

本研究の一部は横浜国立大学人工知能研究拠点学長裁量経費の支援による．

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, 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.

[3] 能條太悟, 杉本航洋, 富井尚志. SQL ライクな操作言語を用いた可視化システムの応用とライフログ分析のためのデータ操作高速化. 日本データベース学会和文論文誌, 第 22-J 巻, pp. 1–10, 2023.

[4] 木次輝, 宇津木萌, 大久保勇輝, 富井尚志. SQL を拡張した情報可視化言語を用いたライフログデータ分析支援システムのパフォーマンス評価. 第 17 回データ工学と情報マネジメントに関するフォーラム (DEIM2025), 9G-04, pp. 1–8, 2025.

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

[6] 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.

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

[8] 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.

[9] Wenqiang Cui. Visual analytics: A comprehensive overview. *IEEE Access*, Vol. 7, pp. 81555–81573, 2019.

[10] 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.

[11] 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.

[12] Jinwook Bok, Bohyoung Kim, and Jinwook Seo. Augmenting parallel coordinates plots with color-coded stacked histograms. *IEEE Transactions on Visualization and Computer Graphics*, Vol. 28, No. 7, pp. 2563–2576, 2022.

[13] Henning Gruendl, Patrick Riehmann, Yves Pausch, and Bernd Fröhlich. Time‐series plots integrated in parallel‐coordinates displays. *Computer Graphics Forum*, Vol. 35, , 2016.

[14] Lena Cibulski, Thorsten May, Johanna Schmidt, and Jorn Kohlhammer. Compo*sed: Composite parallel coordinates for co-dependent multi-attribute choices. *IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS*, Vol. 29, No. 10, 2023.

[15] 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.

[16] 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.

[17] 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.

[18] Fotis Psallidas and Eugene Wu. Provenance for interactive visualizations. HILDA ' 18: Workshop on Human-In-the-Loop Data Analytics, 2018.

[19] 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.

[20] Holger Stitz, Samuel Gratzl, Harald Piringer, Thomas Zichner, and Marc Streit. Knowledgepearls: Provenance-based visualization retrieval. *IEEE Transactions on Visualization and Computer Graphics*, Vol. 25, No. 1, pp. 120–130, 2019.

[21] 増永良文. リレーショナルデータベース入門―データモデル・SQL・管理システム. サイエンス社, 2003.