SuperSQLを用いた非手続き的な記述によるインクリメンタルwebページの生成

田嶋　将大† 　五嶋　研人† 　遠山元道††

† 慶應義塾大学理工学部情報工学科　〒 223-8522 神奈川県横浜市港北区日吉 3-14-1
E-mail: †{taji,goto}@db.ics.keio.ac.jp, ††toyama@ics.keio.ac.jp

あらまし　近年多様化している web ページ上におけるデータの取得およびその提示方法の内, 特に注目を集めているインクリメンタルに取得・提示を行う web ページの実現には非同期通信を可能にする Ajax を使用した実装方式がメジャーであり, その実装をサポートするプラグインが近年多く公開されている. しかし, web 開発者によるデータ取得の為のサーバーサイドの実装が煩雑になってしまう問題点がある. そこで本研究では, 関係データベースの出力結果を構造化し, 多様なレイアウト表現を可能とする SQL の拡張言語である SuperSQL を使用した非手続き的な手法によるインクリメンタルな Web ページ生成の簡略化を目指している.

キーワード　データベース, SQL, SuperSQL, インクリメンタル web, Infinite-Scroll, トリガ, プロシージャ, ログ, バッファリング

1. はじめに

近年 web ページ上におけるデータの取得およびその提示方法が多様化してきている. その一つに, インクリメンタルにデータを取得・提示を行うインクリメンタルな web ページがある. このインクリメンタルな web ページの例として, 無限スクロールと呼ばれる web ページの機能がある. 無限スクロールを用いた web ページでは, 一定の順に追加データの取得・提示が行われ, ユーザのスクロール操作による閲覧方向は垂直または水平方向に固定されている. また近年では Twitter や Facebook などのソーシャルメディアサイトを始めとして多くの web サイトで使用されている. このインクリメンタルな web ページの実現には, 非同期通信を可能にする Ajax [1] を使用した実装方式がメジャーであり, その実装をサポートするプラグインが近年多く公開されている. しかし, これらのプラグインは web 開発におけるクライアントサイドの実装の簡易化を目的としている一方で, データ取得の為のサーバーサイドプログラミングの実装は web 開発者が全て行う必要があり, その実装が煩雑になってしまう問題点がある.

また, データを一括して取得・提示を行う web ページの開発手法の一つに, 遠山研究室で独自に研究開発が進められている SuperSQL がある. この SuperSQL は, 独自のクエリを記述することによって関係データベースの出力結果を構造化し, 多様なレイアウト表現を可能とする SQL の拡張言語であり, 非手続き的な web ページの開発を可能としている. この SuperSQL では, クエリに記述された属性のデータを全て一括して取得する為, 上記のようなインクリメンタルにデータを取得する機構の実現ができず, インクリメンタルな web ページの実現ができていなかった. そこで本研究では, SuperSQL を使用した非手続き的な手法によるインクリメンタルな Web ページ生成の簡略化を目指している. 提案実現の為, 1 ソースから複数のデータコンテンツ取得用のスクリプト自動生成を実現する新たなレイアウト構造生成機構を開発した他, 従来の SuperSQL では実現できていなかった, インクリメンタルなデータ分割取得の為のデータ取得機構の拡張を行った.

また, 本提案手法ではインクリメンタルなデータ提示の実現の為, 取得したデータをクライアントサイドでバッファリングを行い, バッファリングデータからの web コンテンツの生成を実現している. 追加バッファリングを行う際, データベースにおける delete や update による更新によってデータの取得漏れを起こす可能性がある. そこで, 追加データ取得の精度向上を目的とし, トリガ・プロシージャとログテーブルを用いた追加取得時の取得開始位置の特定を提案している.

以降, 本稿では第 2 章で SuperSQL について, 第 3 章では関連研究・関連技術について, 第 4 章で SuperSQL によるインクリメンタル web ページの生成について, 第 5 章で実験・評価について述べ, 第 6 章でまとめを記述する.

2. SuperSQL とは

2.1 SuperSQL クエリ

SuperSQL は関係データベースの出力結果を構造化し, 多様なレイアウト表現を可能とする SQL の拡張言語であり, 慶應義塾大学遠山研究室で開発されている [2] [3]. そのクエリは SQL の SELECT 句を GENERATE <*media*><*TFE*> の構文を持つ GENERATE 句で置き換えたものである. <*media*> は出力媒体を示し, HTML, PDF, Mobile_HTML5 [4] などの指定ができる. また <*TFE*> はターゲットリストの拡張である Target Form Expression を表し, 結合子, 反復子などのレイアウト指定演算子を持つ一種の式である.

2.2 埋め込み型 SuperSQL

埋め込み型 SuperSQL は [5] [6], 動的 web 開発言語である SuperSQL を PHP の関数群として実現することにより, 近年

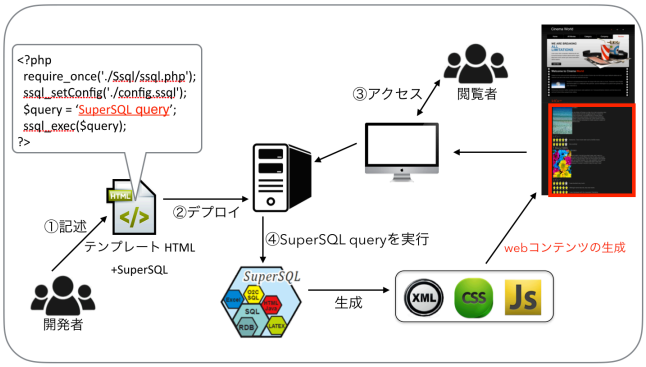


図 1　埋め込み型 SuperSQL アーキテクチャ

web 上で容易に入手することができるフレームワークなどを利用したリッチなデザインの HTML 田プレートを含む既存のコードに対して, SuperSQL のクエリの PHP 経由での埋め込みを可能としている.

埋め込み型 SuperSQL では, 図 1 に示すように, データを XML で受け渡し, JavaScript で実装された SuperSQL レンダリングエンジンを用いて web コンテンツのレンダリングを行う, クライアントサイド処理機構により, 生成した web コンテンツのメンテナンス性の向上を図っている.

3.　関連研究・関連技術

3.1　関 連 技 術

3.1.1　Web アプリケーション開発

Web アプリケーションは, 主に PHP [7], Ruby [8], Perl [9] などのスクリプト言語を使用して開発されている. これらの言語は, ブログ, ソーシャルネットワーキングサービス, オンラインショッピングプラットフォームなどのウェブサイトを作成するために広く使用されている. Web アプリケーションの開発には, プログラミング言語, HTML, CSS, JavaScript を使用したフロントエンド web 開発, およびデータベースの操作に関する知識が必要となる. Web 開発をサポートする為, web アプリケーションフレームワークが多く提供されている.

3.1.2　Web アプリケーションフレームワーク

Web アプリケーションフレームワークは, web サービスや web アプリケーションの開発用に設計されており, このフレームワークの目的は, 開発者にとって web サイトの開発をより迅速かつ容易にすることである. 機能はフレームワークによって異なるが, ほとんどのフレームワークでは, ある種のテンプレートやライブラリが採用されている. 現在, さまざまなプログラミング言語用の web アプリケーションフレームワークが多数存在している. たとえば, PHP の場合, CakePHP [10], Lavarel [11], Biscuit [12], および Symfony [13] が広く使用されている. Play Framework [14], Tapestry [15], Velocity [16] は Java 用, Ark [17], Catalyst [18], Mojolicious [19] は Perl 用であり, Django [20], TurboGears [21], Flask [22] は Python 用である. また, アプリケーションの作成を支援する JavaScript ライブラリである react [23] と angular [24] も注目されている. 無限スクロールの実装をサポートする目的でこれらのライブラリを拡張したさまざまなライブラリが開発されている. ただし, これらのライブラリはデータベースへのアクセスをサポートしていないため, データベースへのアクセスを実装するには, 開発者が Ajax 通信機能を実装し, サーバー側のプログラムを実装する必要がある為, JavaScript およびサーバーサイドプログラムの深い知識が必要となり, 手続き的で複雑なコードディングを行う必要がある. ほとんどすべてのフレームワークは, オブジェクトリレーショナルマッピングを採用してデータベースにアクセスする. オブジェクト・リレーショナル・マッピングでは, 開発者は, 基礎となるリレーショナル・データベースにマップされたオブジェクト・セットのコンテンツを含む HTML ファイルを出力する手続き型ロジックをコーディングする必要がある. しかし, SuperSQL では, クエリの TFE 内で非手続き的なレイアウト構造に従って HTML ファイルを自動的に生成する.

3.2　非同期通信によるインクリメンタル取得

ここでは, 近年の web ページ開発において注目を集めている無限スクロールと呼ばれる web ページにおける機能とその実現手法について述べる. 無限スクロールは, google map の登場から注目を集めてきた, JavaScript と DOM を用いた非同期通信を行う Ajax を基盤として実現がなされている. 無限スクロールでは, 非同期通信を用いることにより, 大量のデータの中からごく一部のデータのみを表示させることができる為, ページの読み込みの時間を削減することができることに加え, 閲覧者が要求した際にのみインクリメンタルにデータを取得することができる為, データベースからのデータ転送の負荷を削減することができる. これらの無限スクロールの特性は, ソーシャルメディアサイトなどのように, 古いデータなどの閲覧者からの要求回数が少ないと予想されるようなデータを持つような, ロングテールな性質を持ったデータセットで使用した場合に大きな効果を持つと考えられる. また, この無限スクロールによる UI に関する様々な研究が行われている. 例えば、Zhang ら [25] は Infinite-scroll を用いたウェブサイトのユーザインタフェースについて様々な研究を行っており, Pinterest のようなソーシャルメディアサイトでのユーザーの投稿に対する無限スクロールの使用の影響の分析を行った. また, Pingfei ら [26] は, 電子商取引サイトの検索結果画面上でページネーションやスクロールで Web サイトを閲覧した場合の検索効率と商品認識への影響を測定した. また, Jaewon ら [27] は, モバイルデバイスで web 検索を実行する際のページネーションと無限スクロールの有効性の比較を行った.

この無限スクロールの実装のサポートを目的とし, 近年では多くのプラグインが公開されており, Infinite Scroll [28] や jScroll [29] などでは, 無限スクロールを適応する HTML タグの指定や追加データ取得時に呼び出されるサーバーサイドプログラムの指定などのクライアントサイドの実装を非手続き的な手法として実現しており, web 開発者のクライアントサイドの実装コストの削減を実現している. しかし, こういったプラグインを用いた場合も, web 開発者はデータ取得用のサーバーサイドプログラムの実装を行う必要があり, また, 各プラグインにより無限スクロール実現の為に必要な HTML タグの構造が異

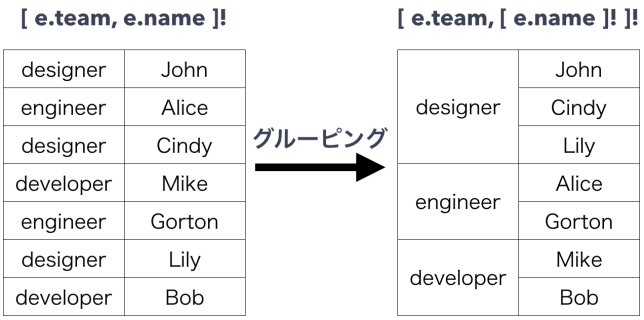


図 2　グルーピング

なる為, 使用するプラグインの機能を考慮した HTML タグの構造の実装を行う必要があり, その実装が煩雑になってしまうといった問題点が挙げられる.

4. SuperSQL によるインクリメンタル web ページの生成

ここでは, 本提案手法によるインクリメンタル web ページの生成について述べる.

4.1 クエリによる無限スクロールの指定

本提案手法では, SuperSQL の反復子に対し装飾子@{infinite-scroll=n} を指定することにより, 右辺に指定された指定個数ずつデータを表示するインクリメンタルな Web ページの生成を行う. また, SuperSQL クエリでは, 反復子の入れ子構造を持たせることにより, 属性の値の関係によるグルーピングを生成することができる. 図 2 のように, 外側の反復子の要素の値によって, 内側の反復子の要素をグルーピングすることができる. このような入れ子構造を持った反復子に対しても装飾子@{infinite-scroll=n} を指定することができ, 1 クエリで 1 ページ内に複数の無限スクロールを適応させた web ページを生成することができる. sample1.ssql のようなクエリによって, 図 3 に示すように TFE1 の値が n1 個ずつに分割して縦方向に表示され, 各 TFE1 の値に関連した TFE2 の値が n2 個ずつに分割して横方向に, TFE4 の値が n4 個ずつに分割して縦方向にそれぞれ表示され, また, 同様に各 TFE2 の値に関連した TFE3 の値が n3 個ずつに分割して縦方向に表示されるような, 入れ子構造に無限スクロールを適応した web ページが生成される.

sample1.ssql

```
[ TFE1 ,
    [ TFE2 !
        [ TFE3 ]!@{infinite-scroll=n3}
    ],@{infinite-scroll=n2},
    [ TFE4 ]!@{infinite-scroll=n4}
]!@{infinite-scroll=n1}
```

4.2 アーキテクチャ

図 4 および図 5 に提案手法のアーキテクチャを示す. 図 4 に示すように, web ページ開発者は埋め込み型 SuperSQL を用いて, HTML テンプレート内に SuperSQL クエリの埋め込みを行うことによってインクリメンタル web ページの開発を行う. また, 該当ページに閲覧者がアクセスした際に SuperSQL クエリが実行され, PHP, CSS, JavaScript が生成される. 生成された PHP によって, データベースにデータの問い合わせが行われ, XML 形式のバッファデータが生成される. 閲覧者には SuperSQL クエリに指定されている指定個数の web コンテンツがこのバッファデータから生成され表示される. また図 5 に示すように, 該当ページ内において閲覧者のスクロール操作により追加コンテンツの要求が行われた場合, 既に生成済みのバッファデータから追加 web コンテンツの生成が行われる. また, 全てのバッファデータが使用済みの場合は再度データベースへのデータの問い合わせを行い, 追加バッファデータの生成を行う.

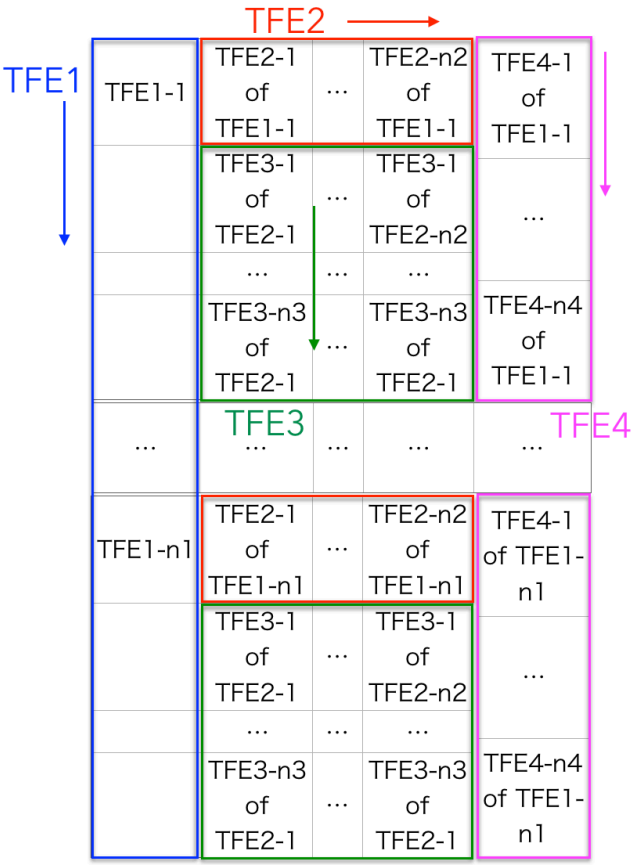


図 3　sample1.ssql によって生成される web ページのイメージ

4.3 分割バッファリング機構

ここでは, SuperSQL クエリから分割バッファリングを実現する為の SQL クエリの生成について, およびバッファデータ生成時のデータ取得方法について述べる.

4.3.1 SQL クエリの分割生成

従来の SuperSQL では, 一括取得を行う web ページの生成のみを対象としていた為, sample2.ssql のような構造を持ったクエリから, sample2.sql に示されるように, SuperSQL クエリ内に記述されている属性全てのデータを一括して取得するような SQL クエリが生成され, 得られるデータを, クエリで示されるレイアウトに従い構造化を行い, 全ての web コンテンツの生成を行なっていた. しかし, SuperSQL によってインクリメンタル web ページの生成を行う場合, 従来の SuperSQL によって生

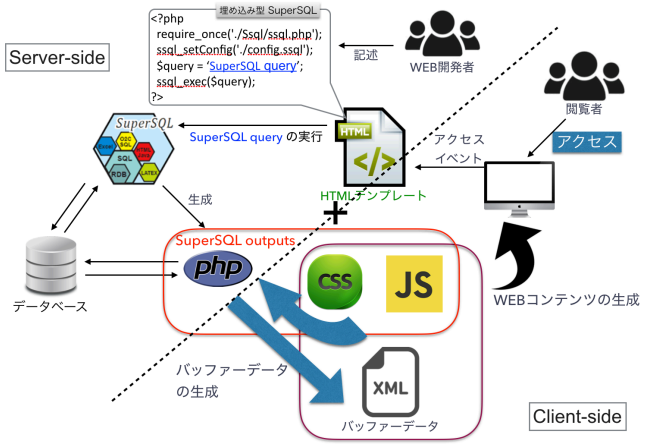


図 4　アーキテクチャ：Web ページ開発–アクセス時

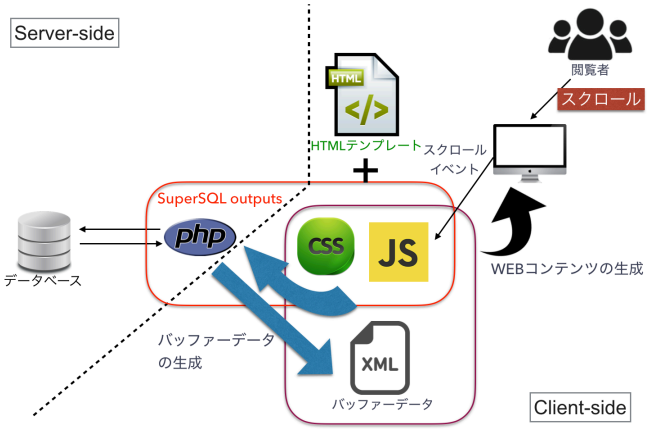


図 5　アーキテクチャ：スクロール操作時

成される SQL クエリでは, データベースから全てのデータを取得することとなり, インクリメンタルなデータ取得とは言えず, インクリメンタル web ページの利点を活かした web ページの生成は実現できないと考えられる.

また, ある反復子において, グルーピングが行われている場合には, そのグルーピングにおけるキーとなる要素をインクリメンタルに取得することになる為, 従来のように 1 つの SQL クエリのみでは, 無限スクロールによる追加データの取得が適切に行えないことや, 1 ページ内の複数の無限スクロールによる各要素のインクリメンタルな取得に対応することができないという問題がある.

そこで, 本提案システムでは, 1 つの SuperSQL クエリから, 複数の無限スクロールによるインクリメンタルなデータの取得の為, SQL クエリの分割生成を行なっている.

sample2.ssql

```
GENERATE PHP
 [ TFE1 ] !
FROM ... WHERE ...
```

sample2.sql

```
SELECT TFE1 FROM ... WHERE ...
```

本提案システムでは, クエリ内の任意の反復子に対し, 装飾子 infinite-scroll を指定可能である為, SQL クエリの分割生成時に, インクリメンタル取得の対象となる属性集合と非対象属性集合の分類を行う. 以下では, 対象属性集合の特定のメカニズムとそれぞれに対して生成される SQL クエリについて説明する.

sample3.ssql

```
GENERATE PHP
 [ TFE1 ,
    [ TFE2 !
       [ TFE3 ]!@{infinite-scroll=n3}
    ],
 ]!@{infinite-scroll=n1}
FROM ... WHERE ...;
```

装飾子 infinite-scroll が指定されている各反復子に対し, グルーピングが行われているかの判定を行う. グルーピングが行われている場合には, グルーピングのキーの役割を果たしている TFE 内の属性全てをその反復子に対する対象属性集合とする. また, グルーピングが行われてない場合には, 反復子内の TEF の属性全てをその反復子に対する対象属性集合とする. また, 各反復子に対し, 対象属性集合として判定されなかった TFE 内の属性全てをその反復子に対する非対象属性集合と判定される. sample3.ssql のクエリから, 図 6 のような構文解析木が構築される. この時, 一番階層の浅い反復子に対して装飾子 infinite-scroll が指定されており, TFE1 をキーとしてグルーピングが行われている為, TFE1 内に含まれる属性全て (att1-1, att1-2, att1-3, ...) が n1 個ずつの分割取得対象属性集合となる. また, TFE1 でグルーピングされている反復子に対しては装飾子 infinite-scroll が指定されていない為, TFE2 内に含まれる属性全て (att2-1, att2-2, att2-3, ...) が非対象属性集合となる. さらに, 一番階層の深い反復子に対して装飾子 infinite-scroll が指定されているが, グルーピングがされていない為, 反復子内で使用されている属性全て (att3-1, att3-2, att3-3, ...) が n3 個ずつの分割取得対象属性集合となる.

以上のように分類を行ったそれぞれの対象属性集合および非対象属性集合のデータ取得の為, それぞれ SQL クエリが生成される. 対象属性集合用の SQL クエリでは, それぞれインクリメンタルな取得を実現する為, limit-offset を使用したクエリの発行を行う. この時, limit 値はバッファリングを行う為, 指定された分割表示数 ∗ 10 に設定している. また, 非対象属性集合にはそれぞれ一括取得を行う為, limit-offset を使用しないクエリが発行される.

生成される対象属性集合用 SQL クエリ

```
SELECT DISTINCT att1-1, att1-2, att1-3, ... FROM
... WHERE ... limit n1*10 offset $1;
SELECT DISTINCT att3-1, att3-2, att3-3, ... FROM
... WHERE ... limit n3*10 offset $1;
```

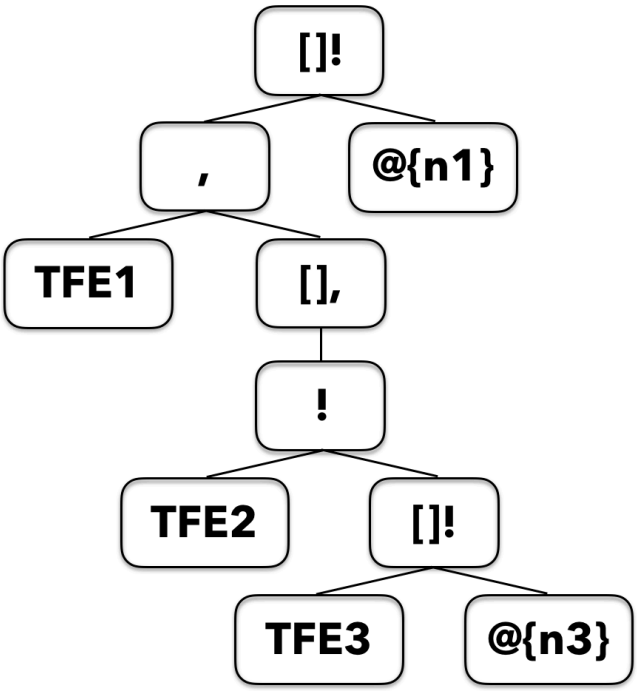


図 6　ツ　リ　ー

生成される非対象属性集合用 SQL クエリ

SELECT DISTINCT att2-1, att2-2, att2-3, ... FROM ... WHERE ... ;

4.3.2　分割バッファリング機構

提案手法では, 4.3.1 で述べたように生成された SQL クエリによって, インクリメンタルにデータの取得を行いバッファデータの生成を行なっている. ここでは, 本論文で提案する 2 種類のバッファリング手法について述べる.

表 1　バッファリング手法

	スナップショット モード	プリサイス モード
データ取得精度	取得漏れの可能性高	取得漏れの可能性低
コスト	オーバーヘッド低	オーバーヘッド高

表 1 に 2 種類のバッファリング手法の概要を示す. スナップショットモードは, 装飾子 infinite-scroll と一緒に"buffer-manage='snapshot' " の記述を行うことによって指定することが可能であり, buffer-manage の指定がなかった場合もスナップショットモードが使用される. スナップショットモードでは, 4.3.1 で生成された分割取得用の SQL クエリの offset 値を *limit* 値 * 追加データ要求回数 に設定する為, 追加データの取得開始位置の特定によるコストは低い一方で, データベースにおける delete や update などによるデータベースの更新を一切考慮せずデータの取得が行われる為, タプルの追加削除が頻繁に行われるデータベースでは, データの取得漏れを起こす可能性が高くなってしまうといったデメリットがある. 一方, プリサイスモードでは, 装飾子 infinite-scroll と一緒に"buffer-manage='precise' " の記述を行うことによって指定することができる. 後述するバッファデータとログテーブルを用いてデータベースにおける delete や update などによるデータベースの更新を考慮し, 分割取得用の SQL クエリの offset 値の算出を行う. その為, プリサイス モードでは, バッファデータ数が増えるとその offset 値の計算コストが高くなってしまうといったデメリットがあるが, その分, スナップショットモードに比べ, データの取得漏れの可能性を下げることを目的としている. このプリサイスモードでは, SuperSQL 実行時にログテーブルの自動生成と同時に, データベースにおける更新を検出しその結果を必要の応じてログテーブルに挿入を行うトリガ・プロシージャの自動登録を行っている. 図 7 にスナップショットモードとプリサイスモードの比較を示す.

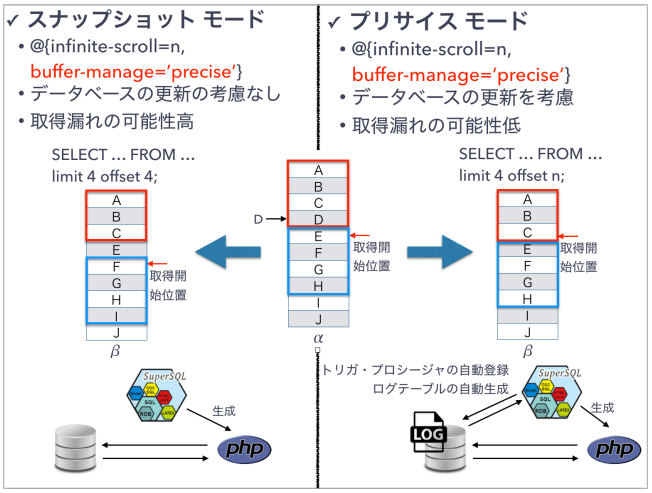


図 7　スナップショットモードとプリサイスモードの比較

Algorithm1 はプリサイスモードにおける, SuperSQL 実行時のログテーブルの自動生成およびトリガ・プロシージャの自動登録の擬似コードを表している.

Algorithm 1 ログテーブルの自動生成およびトリガ・プロシージャの自動登録

$ATT \Leftarrow$ ∀ 対象属性集合//*e.g.* [$\{att1-1, att1-2, att1-3, ...\}, \{att3-1, att3-2, att3-3, ...\}$]
$SIZE \Leftarrow ATT.size()$
$index \Leftarrow 0$
$F \Leftarrow FROM$ 句内のテーブル集合
$W \Leftarrow WHERE$ 句内に使用されているテーブル集合
while $index \neq SIZE$ **do**
  $atts \Leftarrow ATT.get(index)$//*e.g.*$\{att1-1, att1-2, att1-3, ...\}$
  $create\_log\_table()$
  $T \Leftarrow atts$ 内の属性に対応するテーブル集合
  $R \Leftarrow \{t' \in W \mid \forall\{t1, t2\} \subset T$ の関連関係であるテーブル $\}$
  $\forall t \in (T \cup R)$ にトリガ・プロシージャを登録
  $index++$
end while

4.3.1 で分類された各対象属性集合に対し, ログテーブルの生成を行う. 次に, FROM 句内の記述から, 対象属性集合内の各属性に対応するテーブルを算出しそのテーブルの集合を T とする. 続いて, T 内の任意の 2 つのテーブルの関連関係である

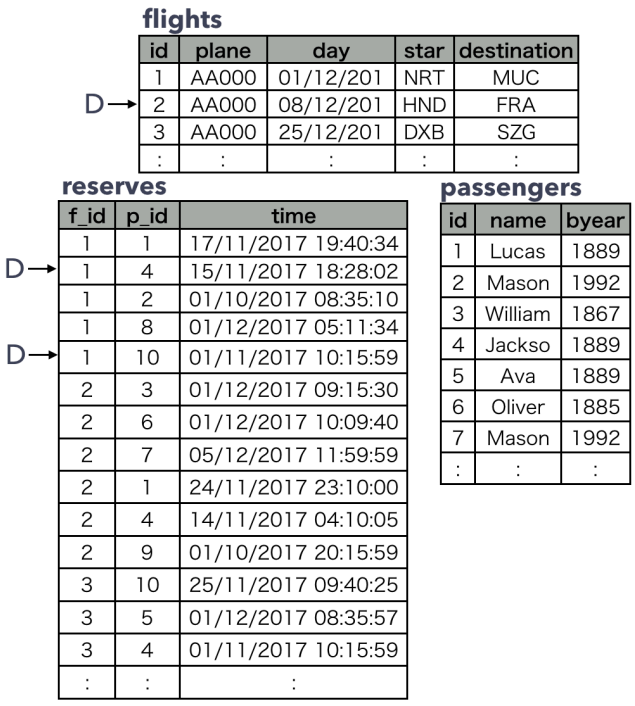


図 8　フライトのサンプルデータ

テーブルを WHERE 句の記述から算出を行い, 関連関係であるテーブルの集合を R とする. これら T と R の和集合 $T \cup R$ 内の各テーブルに対し, トリガ・プロシージャの登録を行う. ここで生成されるログテーブルは, 対象属性集合内の属性のデータの値の組み合わせの内, データベース上に存在しなくなった値の組み合わせのみを保管するテーブルである. また, 登録された, トリガは各テーブルに対する delete と update による更新を検出し対応するプロシージャの呼び出しを行い, プロシージャは, 対象属性集合内の値の組み合わせを確認し必要に応じてログテーブルに挿入を行う.

ここで, 説明の為, 飛行機のフライトとその予約を対象としたデータベースを例に挙げる. 図 8 はフライトのサンプルデータベースの一部のテーブルとそのデータを表している. このようなデータベースに対し, sample4.ssql のようなクエリを用いた場合, 一番外側の反復子に対してのみ装飾子 infinite-scroll が指定されており, その反復子内では f.flight, f.day, p.byear をキーとしてグルーピングがされている為, 対象属性集合は {f.flight, f.day, p.byear} となる. また, reserves テーブルは対象属性集合内の属性に対応する flights, passengers テーブルの関連関係である為, flights, passengers テーブルと合わせて reserves テーブルに対してもトリガ・プロシージャの登録が行われる.

この時, 図 8 のように, flights, reserves テーブルにおいて計 3 つのタプルが delete されることを想定する. flights, reserves テーブルに登録されているトリガはこれらのタプルの削除を検出し, 対応するプロシージャの呼び出しを行う. この時, 対象属性集合の値の組み合わせは, 図 9 の result のようになり, 8 つのタプルが削除されたことと解釈することができる. トリガによって呼び出されたプロシージャはこの result の中に存在しな

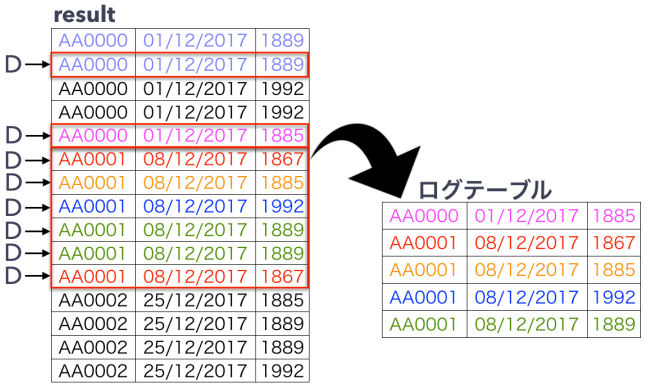


図 9　対象属性集合の検索結果およびログテーブル

くなった値の組み合わせを持つタプルのみをログテーブルに挿入する. 従って, {AA0000, 01/12/2017, 1889} の値の組み合わせを持つタプルは, 削除されているものの result 内には残っていることがわかる為, この値の組み合わせはログテーブルには挿入されない.

sample4.ssql

```
GENERATE PHP
 [  f.plane, f.day!  p.byear!
    [  p.name  ],
 ]!@{infinite-scroll=5}
FROM flights f, reserves r, passengers p
WHERE f.id = r.f_id and p.id = r.p_id;
```

4.3.3　ログテーブルを用いた取得開始位置の特定

ここでは, 4.3.2 で生成したログテーブルおよび生成済みのバッファデータを用いた, offset 値の算出による取得開始位置の特定について述べる.

Algorithm 2 ログテーブルとバッファデータを用いた取得開始位置の特定

```
B ⇐ {b_1, b_2, ...b_N}//バッファ済データ
num ⇐ B.size()//num = N
offset ⇐ num
index ⇐ 0
diff ⇐ 0
while index ≠ num do
  values ⇐ B.get(index)
  if values ⊂ LogTable then
    diff++
  end if
  index++
end while
offset ⇐ offset − diff
```

Algorithm 2 に取得開始位置の特定アルゴリズムの擬似コードを載せる. バッファ済データ B は対象属性集合のバッファ済のデータの値の組み合わせを格納する配列を表し, 各値の組み合わせ (b_i) に対し, その値の組み合わせがログテーブル中に存在するかを確認し, 存在する場合には, バッファ生成後にその値

の組み合わせがデータベースに存在しなくなったと判断することができ, このタプルをバッファデータとデータベースとの誤差としてカウントする. 取得開始位置は, 既に取得したデータの数から上記の誤差を引いた値として算出し, offset 値として使用する.

5. 実験・評価

5.1 コード量評価

本提案手法によるインクリメンタル web ページ作成におけるコスト評価を行う為, 手続き的な他手法を用いた場合と提案手法を用いた場合でインクリメンタル web サイトの作成を行い, そのコード量による評価を行った. 以下の 3 つの手法を比較対象として採用した.

- Ajax : HTML + PHP + CSS + JavaScript
- infinite-scroll : HTML + PHP + CSS + JavaScript + infinite-scroll
- jScroll : HTML + PHP + CSS + JavaScript + jScroll

Ajax では, HTML, PHP, CSS, JavaScript のみを用いて, プラグインなどは使用せず実装を行なった. また, infinite-scroll, jScroll はそれぞれ Ajax による実装と同様, HTML, PHP, CSS, JavaScript に加え, プラグインを使用した実装を行なった.

また, それぞれの手法を用いて, 評価用 web サイトとして, 以下の 2 つ種類の web サイトの実装を行なった.

（1） Facebook を模範にした web サイト

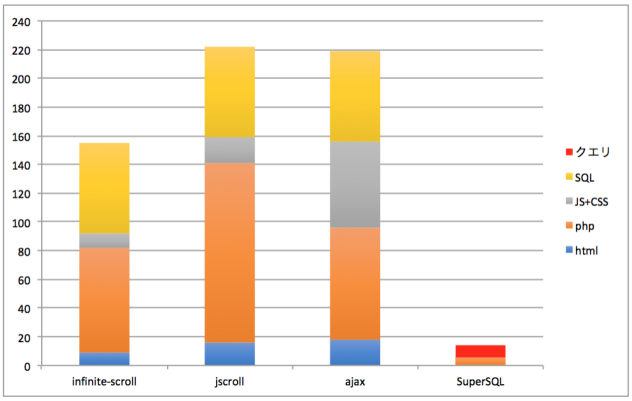
（2） slack を模範にした web サイト

サイト (1) は, ユーザーによる投稿およびそれに対応したコメントを 1 ページ内で表示し, それぞれに無限スクロールを適応させた web サイトであり, サイト (2) は, 無限スクロールを適応させないチャンネル一覧ページと無限スクロールを適応させた, 各チャンネル内での投稿内容を表示させるページの 2 つのページでできる 1 つの web サイトである. 両サイトとも, 本論文で提案している, ログテーブルとバッファを用いた追加データ取得時の取得開始位置特定を採用している.

図 10 はコード量の比較結果を表す. 図 10 から, 提案手法は他手法に比べ, 6〜15%のコード量を実現していることがわかる. 特に, 実装が複雑かつ膨大になりがちな PHP を自動生成することで, web 開発者の PHP の記述量を大幅に抑えることができたこと, また, 無限スクロール適応のために必要な JavaScript の記述や, トリガ・プロシージャの登録などに必要な SQL の記述も無くすことができたことによって, コストの削減を実現できていることがわかる.

5.2 インクリメンタルなバッファリング機構の性能評価

インクリメンタルなバッファリングの手法として提案している, スナップショットモードとプリサイスモードの性能評価の為, 評価実験を行なった. 実験内容としては, トランザクション発生間隔とトランザクションの発生回数をパラメータとし, delete または update を行うトランザクションを発生させ, それぞれの環境下で, 閲覧者には一定のスピードを保ちつつ web ページ上でスクロールによる追加 web コンテンツの要求を行なってもらった.



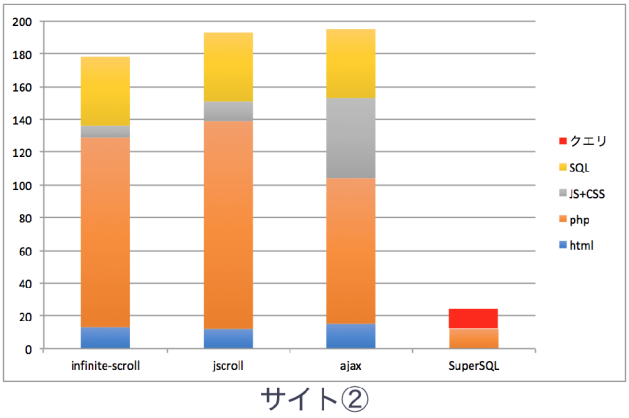


図 10 コード量比較結果

性能比較の基準として, 取得成功比を使用する. これは, 閲覧開始時にデータベースに存在する取得予定のデータの数 (All) に対する, 閲覧終了時までに実際に取得されたデータの数 (Buffer) の割合を表しており, 式 1 で計算される値を使用している.

$$\text{取得成功比} = \frac{Buffer}{All} \tag{1}$$

実験結果は図 11 のようになった. 図 11 から, プリサイスモードがスナップショットモードよりもトランザクションの発生頻度, 発生回数に依らず, 常に正しくデータの取得を行うことができていることがわかる. 従って, ログテーブルとバッファデータを使用した取得開始位置の算出が有用であることがわかる.

また, 図 11 下からわかるように, トランザクション頻度の低いような環境では, その発生回数が上がっても, 90%以上の取得成功比を保っていることがわかる. しかし, 図 11 上から, トランザクションの発生頻度が上がると, プリサイスモードにおいてもその取得成功比が落ちてしまうことがわかる. これは, 取得開始位置の特定処理時に別のトランザクションが発生した場合に, そのトランザクションによる影響を取得開始位置の特定に考慮することができていないことが原因として考えられる. また, まだ取得していないデータが削除された場合に取得できるそもそものデータ数が減ってしまったためであると考えられる.

6. ま と め

本論文では, ソーシャルメディアなどの導入によって注目を集

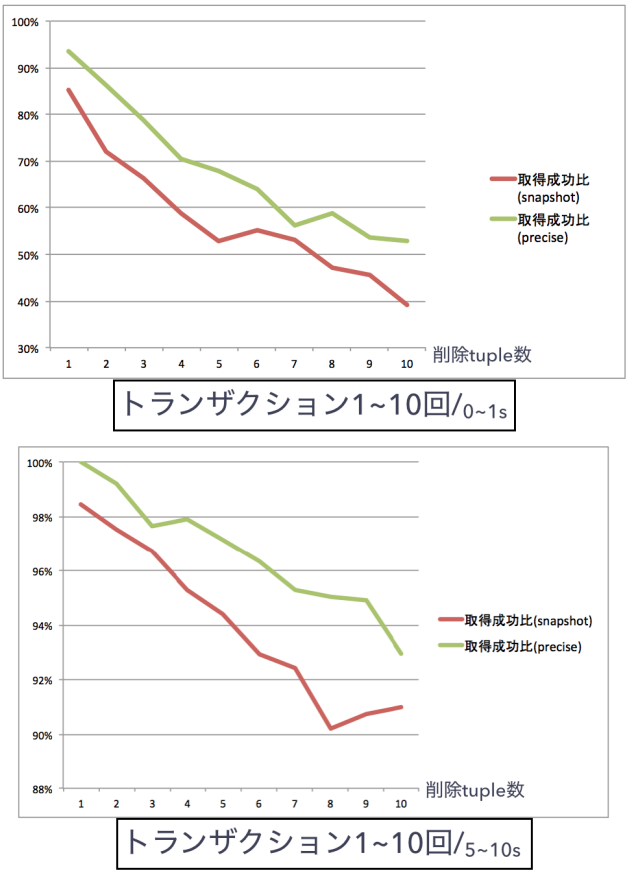


図 11　スナップショットモードおよびプリサイスモードの取得成功比

め, 近年多くの web サイトで採用されているインクリメンタルな web サイトを, 非手続き的な記述のみによる自動生成を提案した. これにより, 従来 Ajax を用いて手続き的に実装を行なっていた為に, 実装が煩雑になってしまっていたインクリメンタル web ページ開発の簡易化を実現した. また, 従来の SuperSQL では全てのデータを一括して取得し構造化を行うことによって web コンテンツを生成していた為, 閲覧者の見る可能性の低いデータを持つようなロングテールな性質を持ったデータセットにおける無駄なデータ転送が生じていたが, インクリメンタルなデータ取得の為の SQL 分割生成機構の導入により, 無駄なデータ転送を防ぐことができる. また, インクリメンタルなデータ取得では, データベースにおける更新によるデータの取得漏れの発生が問題として考えられるが, 追加データ取得時のログテーブルを用いた取得開始位置の特定を行うことにより, 取得漏れの可能性を下げることを実現した.

今後の課題として, 本研究では, 追加データ取得時には, データベースにおける delete または update のみを考慮した取得位置の特定を行なっているが, insert された新しいデータの取得は無限スクロールとは異なる機能として利用されている為, insert を考慮できていない. インクリメンタルな web ページとしては, insert を考慮したデータの取得を行えるようになることで, 株価サイトなどで見られるような, 最新のデータを随時取得し表示が変わるような web ページの自動生成も可能になると考えられる.

文　献

[1] Jesse James Garrett, et al, "Ajax:Anewapproachtowebapplications", 2005
[2] SuperSQL, http://ssql.db.ics.keio.ac.jp, 2013
[3] M. Toyama, "SuperSQL: An Extended SQL for Database Publishing and Presentation", in Proceedings of ACM SIGMOD '98 International Conference on Management of Data, pp. 584-586, 1998.
[4] K. Goto and M. Toyama, "Mobile Web Application Generation Features For SuperSQL", in Proceedings of the 20th International Database Engineering & Applications Symposium, IDEAS 2016, pp. 308-315, 2016.
[5] Masato Kiya, Kento Goto, and Motomichi Toyama, "GENERATE eHTML: Embedding SuperSQL queries in HTML", in Proceedings of the 19th International Database Engineering & Applications Symposium. ACM, 224-225, 2015.
[6] Kento Goto, Masato Kiya, and Motomichi Toyama, "Development of SuperSQL Embedding Mechanism for Web Development Framework", TOD, 2017
[7] PHP.net, http://php.net/, 2001
[8] Ruby: A Programmer 's Best Friend, https://www.ruby-lang.org/, 2006
[9] The Perl Programming Language, https://www.perl.org/, 2002
[10] CakePHP : Build fast, growsolid, https://cakephp.org/, 2005
[11] Laravel - The PHP Framework For Web Artisans, https://laravel.com/, 2015
[12] rainner/biscuit-php, https://github.com/rainner/biscuit-php, 2016
[13] Symfony : High Performance PHP Framework for Web Development, https://symfony.com/, 2005
[14] Play Framework - Build Modern and Scalable Web Apps with Java and Scala, https://www.playframework.com/, 2013
[15] Apache Tapestry Home Page, http://tapestry.apache.org/, 2008
[16] The Apache Velocity Project, http://velocity.apache.org/, 2016
[17] Ark, https://metacpan.org/release/Ark, 2013
[18] Catalyst — Perl MVC web application framework, http://www.catalystframework.org/, 2012
[19] Mojolicious - Perl real-time web framework, http://mojolicious.org/, 2008
[20] Django: The Web framework for perfectionists with deadlines, https://www.djangoproject.com/, 2005
[21] TurboGears, http://turbogears.org/
[22] Flask(A Python Microframework), http:// ask.pocoo.org/, 2010
[23] React - A JavaScript library for building user interfaces, https://facebook.github.io/react/, 2017
[24] Angular, https://angular.io/, 2010
[25] Sarah Yixin Zhang, Libo Liu, "Attention trade-off between two types of user contributions: Effects of pinterest-style in nite scroll layouts on creating original sharing and appreciating others ' sharing", 2013
[26] Pingfei Wang, Qian Fei, "Scrolling or Paging: The Impact of Interaction Style on the Search Result Page of Mobile Commerce Website", in International Conference on Human-Computer Interaction, Springer, pp. 454-457, 2013
[27] JaewonKim, PaulThomas, Ramesh Sankaranarayana, TomGedeon, HwanJin Yoon, "Pagination versus scrolling in mobile web search", in Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, ACM, pp. 751-760, 2016
[28] Metafizzy, infinite-scroll, https://infinite-scroll.com/, 2017
[29] Philip Klauzinski, jScroll, http://jscroll.com/, 2011