情報処理学会論文誌　データベース　Vol.10　No.3　26–44 (Oct. 2017)

Web開発フレームワークのための SuperSQL埋め込み機構の開発

五嶋 研人[1,a)]　木谷 将人[2,b)]　遠山 元道[3,c)]

受付日 2017年3月9日, 採録日 2017年5月8日

概要：近年の急速な Web の発展により，動的 Web サイトや Web アプリケーションといった Web サイトの需要が高まっている．このような需要にともない，様々な動的スクリプト言語の出現，多くのフレームワーク等の開発が行われており，簡単にリッチなレイアウトデザインを持つ Web サイトを作成できる環境が整ってきている．多数の動的 Web サイトが世に出ているなかで，公開されている動的 Web サイトの約 8 割が PHP を利用して作成されている．そこで本稿では，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 Web で容易に入手することができるフレームワーク等を利用したリッチなデザインの HTML テンプレートを含む既存コードに対して SuperSQL クエリの埋め込みを可能とする PHP 埋め込み型 SuperSQL についての提案を行う．本提案手法の処理系では，埋め込み型 SuperSQL の言語処理機構，PHP 関数群の処理機構に加えて，現在の Web の主流である分離傾向を取り入れた，データを XML で受け渡し，JavaScript で実装した 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，およびキャッシュを利用した実行の分岐機構を実現した．これらにより，SuperSQL で生成される Web コンテンツをページ内の任意の位置に表示させることができるようになり，従来の手法よりも容易な，データベースを利用した Web ページの開発が可能となった．また，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や，ページを訪問するユーザに対する XML 形式での二次利用向けデータの提供を可能とし，実行の分岐機構により，サーバ負荷の軽減・SuperSQL 処理（Web ページ表示）の高速化を実現した．

キーワード：埋め込み型言語，SuperSQL，動的 Web，PHP，XML，JavaScript，jQuery，キャッシュ

Development of SuperSQL Embedding Mechanism for Web Development Framework

Kento Goto[1,a)]　Masato Kiya[2,b)]　Motomichi Toyama[3,c)]

Received: March 9, 2017, Accepted: May 8, 2017

Abstract: In recent years, the demand of a dynamic Web site and a Web application is increasing. With such demand, various dynamic scripting languages appeared and the environment that many Web developers can make a Web site with the rich layout design easily is being put into place. Approximately 80% of the open dynamic Web sites are using PHP. In this paper, we propose the PHP embedded SuperSQL as functions of PHP which is a dynamic Web programming language for Web development. SuperSQL that is being researched and developed in Toyama Laboratory of Keio University is included in this embedded language. And it enabled to embed SuperSQL queries to an existing code including rich design HTML templates that you can get easily on the Web in recent years. By using this embedded SuperSQL, it enables to develop a Web page using a database easier than the conventional methods. Also, we propose the system to generate Web contents using XML, CSS and JavaScript. It improved the maintainability of the generated Web contents and enabled to offer the XML data for secondary use to a Web user who visits the page. And, the execution branch mechanism using a cache achieved to reduce a server load and accelerate the SuperSQL processing.

Keywords: embedded language, SuperSQL, dynamic Web, PHP, XML, JavaScript, jQuery, Cache

1. はじめに

近年の急速な Web の発展により，動的 Web サイトや Web アプリケーションといった Web サイトの需要が高まっている．このような需要にともない，様々な動的スクリプト言語の出現，多くのフレームワーク等の開発が行われており，簡単にリッチなレイアウトデザインを持つ Web サイトを作成できる環境が整ってきている．多数の動的 Web サイトが世に出ているなかで，公開されている動的 Web サイトのおよそ 82%が PHP を利用して作成されている [1]．PHP は，標準ライブラリが非常に充実しており，ドキュメントも多く，メジャーな言語，かつ初心者が学習するうえで最もハードルの低い言語といわれている．

著者らの研究室で研究開発が続けられている SQL の拡張言語である SuperSQL [2], [3], [4] も様々な Web ページを作成する手法の 1 つである．SuperSQL を利用した Web サイトの開発の大きなメリットは，第 1 に，SuperSQL を用いて作成した少ないコード量のクエリのみで，データベースに格納されているコンテンツを用いた Web サイト開発を行うことができる点である．Web サイト開発者は HTML，CSS 等の記述を行うことなく Web サイトを構築することができるため，HTML や CSS 等の知識がほとんどない人も簡単に利用することができる．第 2 に，少ないコード量の変更で生成する Web コンテンツのレイアウトを多彩に変形することができることである．

しかしながら，現状の SuperSQL による Web 開発の現状の課題点として，まず，現在までに開発されている Web 開発向けの機構である Generate HTML（以下，Standalone SuperSQL（GENERATE HTML））では，データ一体型，すなわちデータ，レイアウト，スタイルをすべて一体型で生成しているという点があげられる．このため，CSS によるデザインの分離，および XML と JavaScript 等を用いたクライアントサイドによるコンテンツ生成といった近年の Web 開発手法との親和性が低い．2 点目の課題点は，SuperSQL によって生成される Web ページ全体のレイアウト，デザインである．近年見られるようなリッチなデザインを持つ Web サイトを作成しようと考えると，SuperSQL によって生成された Web ページに対して，多少なりともテンプレート等を使用した加工が必要不可欠となる．また，3 点目の課題は，Standalone SuperSQL をそのまま動的 Web の開発に利用した場合，Web 利用者によるページアクセスのたびに SuperSQL の実行が発生し，サーバ負荷・Web ページ表示に時間がかかるといった問題が発生する点である．

そこで，シェア率が高く利用するユーザも非常に多い PHP を用いて SuperSQL で生成されるような Web コンテンツを作成することを考える．PHP 等のサーバサイドプログラミング言語を用いて動的 Web サイトの開発を行う手法では，開発者自身による HTML コードの記述が必要不可欠であり，その開発者自身による HTML コードの記述ミスによるエラーで Web コンテンツがうまく表示されないといった問題がある．近年のブラウザはこのようなミスに対し，自動でエラーを修正してくれる機能を持ったものが増えていているものの，修正にともなうブラウザのレスポンスの低下や，脆弱性が高まってしまう等の問題点があげられる．さらに，Web コンテンツのレイアウトを変更したいと考えたとき，SuperSQL とは異なり，多くのコードの変更が必要となってしまう点も課題としてあげられる．

また，従来研究のアプローチを検討してみた場合，XML 等のデータと jQuery を用いてクライアントサイドで Web コンテンツを生成する手法では，レイアウトのために相応のコードを書く必要がある．GUI ツールを用いた Web 開発手法 [12]，テーブルレイアウトを最適化する手法 [13]，プログラムのエラーを修正する手法 [14] は，プログラミング言語の記述ミスをなくすという問題解決には至っていない．また，単体で独自の表記方法を用いて Web アプリケーション全体を作成する Seaside [15] は，既存の HTML ページの一部のみを Seaside を用いて動的 Web にすることはできないといった課題があげられる．

これらの背景から，本研究（以下，埋め込み型 SuperSQL，もしくは Embedded SuperSQL（GENERATE eHTML））では，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 Web で容易に入手することができるフレームワーク等を利用したリッチなデザインの HTML テンプレートを含む既存コードに対して，SuperSQL クエリの PHP 経由での埋め込みを可能とすることを目的とする．そして，SuperSQL によって生成される Web コンテンツをページ内のユーザの任意の位置に表示させることで，Standalone SuperSQL（GENERATE HTML）よりも容易な動的 Web サイトの作成を可能とする．また，データを XML で受け渡し，JavaScript で実装した 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る．ページを訪問するユーザに対しては，XML 形式での二次利用向けデータの提供を可能とする．さらに，実行の分岐機構により，サーバ負荷の軽減・SuperSQL 処

[1] 慶應義塾大学大学院理工学研究科
Graduate School of Science and Technology, Keio University, Yokohama, Kanagawa 223–8522, Japan
[2] 東海旅客鉄道株式会社
Central Japan Railway Company, Nagoya, Aichi 450–6101, Japan
[3] 慶應義塾大学理工学部
Faculty of Science and Technology, Keio University, Yokohama, Kanagawa 223–8522, Japan
[a)] goto@db.ics.keio.ac.jp
[b)] masato@db.ics.keio.ac.jp
[c)] toyama@ics.keio.ac.jp

理（Web ページ表示）の高速化を実現する．

本提案手法は，下記の 1) から 3) の条件を満たす Web サイト・Web アプリケーションのコンテンツ表示部・form 部の作成における使用を対象としている．

1) 関係データベースを使用
2) サーバサイド言語に PHP を使用
3) 表示時・データベース格納時に，データを処理（3.2.5 項に書かれている機能の使用を除く，データの加工やデータに基づいた処理分岐等）せずにそのままの形で用いる

また，上記を満たし開発に本手法を適用可能な Web サイト・Web アプリケーションの例としては下記のようなものがあげられる．

- 大学等のお知らせページ・お知らせ登録フォーム（例：慶應義塾大学の塾生ページ[*1]のお知らせ・イベント等の表示部）
- 投稿ページを含むニュースアプリ（例：Yahoo! ニュース[*2]のヘッドライン表示部・ランキング表示部・コメント報告部）

本稿では，2 章で関連技術・関連研究，3 章で Standalone SuperSQL，4 章で埋め込み型 SuperSQL の言語について，5 章ではその処理系について，6 章ではその用例について述べる．そして，7 章で実験・評価を述べ，8 章でまとめを記述する．

2. 関連技術・関連研究

2.1 関連技術

近年，動的 Web 開発において世界的に用いられているサーバサイドプログラミング言語は，PHP [5]，ASP.NET [6]，Java [7]，static files [8]，ColdFusion [9] である．特に PHP は全世界で広く使用されており [1]，動的 Web サイトの開発では最もメジャーな言語とされている．

Web コンテンツの生成を行う手法として，XML や JSON 等のデータを JavaScript [10] 単体や JavaScript のライブラリである jQuery [11] を用いて解析し，あたかもデータベースからデータを取得しているかのように Web コンテンツをクライアントサイドで動的に生成するといったものも見られる．この手法では，近年の主流である構造（HTML）とスタイル（CSS）の分離のほかに，データも分離して管理することができるため，メンテナンス性が向上するといったメリットや，二次利用のためのデータ抽出も XML や JSON ファイルをダウンロードする形で実現できるといったメリットがある．

2.2 関連研究

Rode らが提案している Click [12] は GUI ツールを用い，あらかじめ用意されたパーツをドラッグ&ドロップすることによって動的 Web サイトを作成する手法である．プログラミングに慣れていないエンドユーザを対象としている．Marriott らは Web コンテンツのテーブルの自動レイアウト化のための最適なテーブルレイアウトを提供するアルゴリズムを提案している [13]．近年では，様々なフレームワークの出現により，画面サイズの幅によってレイアウトが変わるといったレスポンシブなデザインの Web コンテンツの作成が可能となってきているが，彼らは最も幅の広いレイアウトから，徐々に列の幅を狭めていき，テーブルの高さの増加が最も少なくなるようなレイアウトを繰り返し選択する手法を用いて実現している．2.1 節であげた HTML の記述ミスによる多方面への悪影響を避けるために，Hesam らは PHP のプログラムを修正する 2 つのツールを開発した．1 つはプログラムを静的にチェックし，シンプルなエラーを修正する PHPQuick．もう 1 つはテストケースベースで文字列制約を解くことによってプログラムを修正する PHPRepair である [14]．また，Ducasse らが提案している Seaside [15] は Web アプリケーションを作成するためのフレームワークであり，単体で独自の表記方法を用いて Web アプリケーション全体を作成することができる．

SuperSQL の関連研究としては，SuperSQL の言語機能の拡張 [16], [17], [18], [19], [20], [21]，SuperSQL 処理系の改良 [22], [23], [24], [25] がある．SuperSQL を拡張した手法・システムとしては，XML データを HTML 化する XML 整形出力言語である PPX [26]，ユーザの端末環境に動的なレイアウト適応を実現する ACTIVIEW [27] がある．これらの手法では，クエリ単体でメディアの生成を行うことができ，たとえば HTML メディアの生成の場合，HTML タグのミス等による無効な HTML の生成は起きないワンソース，1 言語によるメリットがある．また，先行研究 [28] では，本稿で述べる埋め込み型 SuperSQL の導入部について提案している．

2.3 提案手法の優位性

1 章に示した条件 1)〜3) をふまえ，提案方式，および従来手法を用いた場合の記述可能な処理とそのコストを**図 1** にまとめる．グラフの横軸は実現可能な処理，縦軸はコストを表している．

また，本節の冒頭で述べた対象の Web 開発における優位性を，2.3.1 項から 2.3.3 項に示す．

2.3.1 関連技術・関連研究との比較

サーバサイドプログラミング言語を用いて動的 Web サイトの開発を行う手法では，開発者自身による HTML コードの記述が不可欠である．それに対して提案手法では，1 章に示した条件 1)〜3) に則した Web 開発において，SuperSQL クエリの記述のみで HTML を自動生成する仕様となって

[*1] http://www.gakuji.keio.ac.jp/
[*2] https://news.yahoo.co.jp/

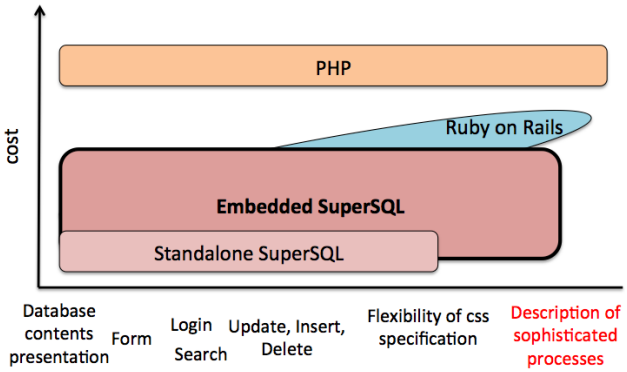

図 1　各言語を用いた場合の記述可能な処理とそのコスト
Fig. 1　Describable processes and those costs when each language is used.

いる．また，提案手法の対象とする Web サイト・Web アプリケーションは，データの加工やデータに基づいた処理分岐（3.2.5 項の機能の使用を除く）等のデータ処理をせずにそのままの形で用いる Web ページとなっているため，1 章に例示したもの等に限定されるが，提案手法は既存手法と比較して少ないコード量で実現できる点に優位性がある．XML 等のデータと jQuery を用いてクライアントサイドで Web コンテンツを生成する手法では，レイアウトのために相応のコードを書く必要ある．Web コンテンツのレイアウトを変更したいと考えたとき，従来手法では，変更内容によっては多くのコードの変更が必要となってしまうが，対象に則した Web 開発において提案手法のクエリは少ないコード量であるため，メンテナンスが容易である点に優位性がある．GUI ツールを用いた Web 開発手法 [12]，テーブルレイアウトを最適化する手法 [13]，プログラムのエラーを修正する手法 [14] は，プログラミング言語の記述ミスをなくすという問題解決には至っていない．提案手法では，少ないコード量によるプログラミングで記述ミスの減少，メンテナンス性の向上を試みた．また，Seaside [15] では既存の HTML ページの一部のみを Seaside を用いて動的 Web にすることは実現していないのに対して，提案手法では，このニーズに対応している．

2.3.2　Web デザイン・テンプレートとの比較

提案手法では，<*T F E*>（3.2 節）内で装飾子（3.2.3 項）を用いることにより，様々なレイアウトの指定が可能となる．提案手法で用意されていない CSS の指定は，装飾子 style を用いて，たとえば，style='font-variant: small-caps; font-weight: 900' といった形で記述することができる（style の右辺に指定できる CSS 記述は，Web における CSS 表記と同じものとなる）．

また，既存 Web デザイン・テンプレートの CSS を提案手法における CSS 生成機構部でも使用したい場合は，装飾子 class（もしくは id）を各属性や反復子（3.2.2 項）に指定することにより対応させることが可能となる．

レイアウトのメンテナンス性については，SuperSQL クエリは，少ない記述の中で指定されているレイアウト情報が把握しやすく，変更が容易である点に優位性がある．

2.3.3　Standalone SuperSQL との比較

Standalone SuperSQL と比較した場合，提案手法は，クライアントサイド処理機構（5.5 節）により，生成した Web コンテンツのメンテナンス性の向上や，ページを訪問するユーザに対する XML 形式での二次利用向けデータの提供を可能としている点において優位性がある．

また，Standalone SuperSQL をそのまま動的 Web の開発に利用した場合，Web 利用者によるページアクセスのたびに SuperSQL の実行が発生し，サーバ負荷・Web ページ表示に時間がかかるといった問題が起こる．それに対して提案手法では，7.3 節の実験から分かるとおり，キャッシュを利用した実行の分岐機構（5.6 節）により，サーバ負荷の軽減・SuperSQL 処理（Web ページ表示）の高速化を実現している点において優位性がある．

3.　Standalone SuperSQL

3.1　概要

Standalone SuperSQL は従来版の SuperSQL のことを指す．本 SuperSQL は SQL を拡張したワンソースマルチユースを実現するクエリ言語である．SuperSQL は，SuperSQL 利用者が用意している DBMS のデータを用いて，既存技術と比較してより少ないコード量でコンテンツ生成を行うことを目標としている．

3.2　クエリ

SuperSQL のクエリ（質問文）は，r_i を関係，P を SQL における条件文とすると以下のように表せる．

GENERATE <*medium*>
<*T F E*>
FROM $r_1, r_2, \ldots, r_n$
WHERE P

ここで <*medium*> は出力媒体を示し，HTML, Excel, SWF 等が指定可能である．また <*T F E* > は SQL におけるターゲットリストの拡張である Target Form Expression を表し，結合子（3.2.1 項），反復子（3.2.2 項）等のレイアウト指定演算子を持つ式である [29]．<*T F E*> 内では，装飾子（3.2.3 項）や関数（3.2.4 項），条件分岐（3.2.5 項）を用いることも可能である．

このクエリをファイル形式（拡張子：ssql）で保存したものをクエリファイルと呼ぶ．SuperSQL 実行時に，このクエリファイルの相対パスを引数（3.4 節）として指定して実行する．

3.2.1　結合子

結合子は，データベースから得られたデータをどの方向

に結合するかを指定する演算子であり，コンマ（,），感嘆符（!），パーセント記号（%）の 3 種類で表す．属性間をこれらの結合子で区切ることによって，それぞれ水平，垂直，深度方向にレイアウトする．出力媒体 *<medium>* に HTML が指定されていた場合，結合子%による結合はハイパーリンクとなる．

3.2.2　反復子

反復子は，指定された方向に，データベースから取得した値を繰り返して表示する．一対の角括弧（[ ]）に結合子（3.2.1 項）を添えたものが反復子となる．

3.2.3　装飾子

装飾子は，関係データベースにより抽出された情報に，文字サイズ，横幅，セル内での文字列の位置，スタイルシートのクラス等の情報を付加する．これらは装飾演算子（@{装飾指定式}）によって指定することができる．装飾指定式は（装飾子名 = 値）として指定する．

3.2.4　関数

関数は，データベース検索結果の文字列に対し，特定の処理を行うための機能である．記述は下記の形式で行う．

関数名(*<TFE>*, *条件文*)

3.2.5　条件分岐

条件分岐による表示内容の変更は，下記のいずれかを用いて行う．*条件式*が真のときは *<TFE 1>* の内容が表示され，偽のときは *<TFE 2>* が表示される．

if(*条件式*) then (*<TFE 1>*) else (*<TFE 2>*)
(*条件式*)? *<TFE 1>* : *<TFE 2>*

3.3　設定ファイル

SuperSQL の実行時に使用するデータベース等の情報は設定ファイルに記述する．その記述は以下のようになる．

driver=*ドライバ名*
host=*接続するデータベースサーバ*
db=*データベース名*
user=*接続アカウント*
password=*パスワード*

SuperSQL は PostgreSQL，MySQL，DB2，SQLite の 4 つの DBMS に対応しており，driver の右辺にそれぞれ postgresql，mysql，db2，sqlite と指定する．

上記の設定ファイルの情報を基にした実行は，SuperSQL 実行時に引数（3.4 節）として指定されたファイルを基にして行う．

3.4　実行引数

SuperSQL では，実行時に指定する引数が用意されている．代表的なものは以下である．

-f *[実行するクエリファイル (3.2 節) の相対パス]*
-d *[生成されるファイル群の出力先]*
-c *[設定ファイル (3.3 節) の相対パス]*
-driver *[ドライバ名]*
-h *[接続するデータベースサーバ]*
-db *[データベース名]*
-u *[接続アカウント]*
-p *[パスワード]*

引数 c を指定しない場合，引数 driver 以下の指定は必須となる．引数 driver 以下を指定している場合，設定ファイル（3.3 節）の作成は不要となる．また，引数 d の指定がない場合，生成されるファイル群の出力先は引数 f で指定されたクエリファイル（3.2 節）が置かれているディレクトリとなる．

4.　埋め込み型 SuperSQL：言語

4.1　概要

本提案手法である埋め込み型 SuperSQL（Embedded SuperSQL）は，既存技術に比べ，より短いコードでの Web コンテンツの生成を実現し，さらに既存技術では大幅なコード量の変更をともなう Web コンテンツのレイアウトの変更も容易に行うことができる非手続き型の Standalone SuperSQL（3 章）の特徴を利用し，HTML テンプレート等の既存コード内に直接 SuperSQL を埋め込むことを目標としている．また，既存コードへの埋め込みにともない，近年の Web の傾向を取り入れ，XML（データと構造），CSS（スタイル），JavaScript（SuperSQL レンダリングエンジン）を用いてクライアントサイドで Web コンテンツを構築する機構を実現した．

また，SuperSQL は Web 開発者が用意している DBMS のデータを用いて Web コンテンツの生成を行うことを前提としており（3.1 節），クエリ（3.2 節）と設定ファイル（3.3 節）に書かれている情報を基にして，提案手法の SuperSQL の処理系（5 章）内でデータベースへのアクセス，XML 等の生成を行い，クライアントサイドではそのデータを Web コンテンツとして表示させる処理を行う仕様となっている．これにより，少ないコード量によるコンテンツ生成を実現している．

4.2　Standalone SuperSQL と提案手法の違い

ここでは，Standalone SuperSQL と提案手法の Embedded SuperSQL の違いについて述べる．**図 2** にその違いを表した図を示す．大きな違いは SuperSQL によって実行されるファイルと生成物である．Standalone SuperSQL では，Web ページ開発者が，作成したクエリファイル（3.2 節）に対して SuperSQL を実行し，生成された Web ページ全体（HTML，CSS，JavaScript 等）を Web サーバに配置，

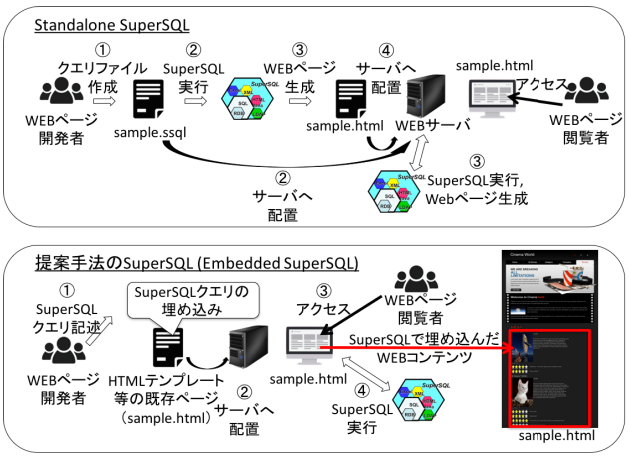

図 2　Standalone SuperSQL と提案手法の SuperSQL の違い
Fig. 2　Difference between Standalone SuperSQL and Embedded SuperSQL.

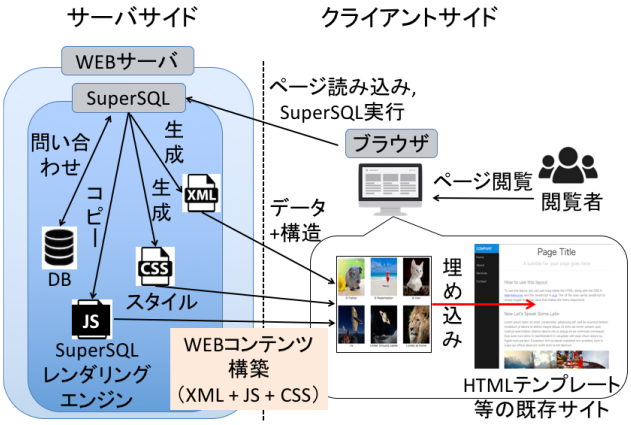

図 3　提案手法のアーキテクチャ
Fig. 3　Architecture of Embedded SuperSQL.

もしくは，作成したクエリファイルを Web サーバへ配置し，サーバ上で SuperSQL が実行され Web ページ全体が生成されるという流れとなっている．一方で提案手法では，HTML テンプレート等の既存コードに対して，Web サイト開発者が SuperSQL のクエリを直接埋め込み，Web サーバへ配置する．そして，該当ページがアクセスされた際に SuperSQL が動的に実行され，Web ページ全体を生成するのではなく，Web コンテンツのみを生成して既存ページ内に直接埋め込んで表示する．提案手法のアーキテクチャを**図 3** に示す．

4.3　PHP 埋め込み型 SuperSQL

提案手法は，Web コンテンツ部分のみを動的に生成してそれを既存コードに埋め込む．そのためには，ページアクセス時に動的に SuperSQL を実行する必要がある．したがって，提案手法では，動的スクリプト言語である PHP に SuperSQL の組み込みを行った．

PHP 上で SuperSQL を動作させるために実装した PHP の関数を**表 1** に示す．ssql_setConfig 関数は SuperSQL の設定（3.3 節）の読み込み，ssql_exec 関数は SuperSQL クエリ（3.2 節）の実行に用いる．

表 1　提案手法で実装した PHP の関数
Table 1　PHP functions of Embedded SuperSQL.

関数名	機能	引数	使用例
ssql_setConfig()	設定読み込み	設定ファイル or 各設定値の指定	ssql_setConfig('./config.ssql') ssql_setConfig('ドライバ名', 'DB 名', 'ホスト名', 'ユーザ名', 'パスワード')
ssql_exec()	SuperSQL 実行	変数（クエリ） or クエリ直接記述	$query = 'GENERATE HTML ...' ssql_exec($query) ssql_exec('GENERATE HTML ...')

表 2　plink 関数と glink 関数の仕様
Table 2　Specification of plink function and glink function.

関数名	値の受け渡し形式	第一引数	第二引数	第三引数以降
plink()	POST 形式	リンクとなる属性	値の受け渡し先	受け渡す値の属性
glink()	GET 形式	リンクとなる属性	値の受け渡し先	受け渡す値の属性

PARAMETER *受け取る値の属性*
GENERATE HTML
<T F E>
FROM *テーブル*
WHERE *条件式*

埋め込み型 SuperSQL では，Standalone SuperSQL の装飾子（3.2.3 項）や関数（3.2.4 項）に加え，新たな関数として，ページアクセス時に SuperSQL が動的に実行されるという性質を利用した plink，glink 関数を実装した．そして，リンク先のページで plink，glink 関数の値の受け取りを行う parameter 句もあわせて実装した．plink 関数，glink 関数の仕様を**表 2** に示す．parameter 句は，下記のように GENERATE 句の前に記述する．リンク元のページのクエリに書かれている plink 関数，もしくは glink 関数により生成されたハイパーリンクを Web 利用者が選択することにより，関数の第 3 引数以降に指定された属性の値がリンク先のページへ受け渡される．そして，その値とリンク先のページに埋め込まれたクエリの parameter 句の情報により，リンク先のページが動的に生成され表示される．リンク先のページに parameter 句を用いたクエリが複数埋め込まれていた場合も，それぞれに値が受け渡されて同様に動的表示が行われる．

また，埋め込み型 SuperSQL では，Web アプリケーションの機能として使用頻度の高い form の機能を実装した．この機能は装飾子 form と**表 3** に示す各装飾子により構成される．form は *<T F E>* @{form} の形で記述され，*<T F E>* 内では，form の構成のために表 3 の各装飾子を用いる．指定された装飾子により，たとえば，入力必須項目（@{notnull}）が入力済みかどうか，数値入力項目（@{number}）に数値以外の値が入力されていないか等のバリデーションが行われ，誤りがある場合には適切なエラーメッセージが表示される．バリデーション後，登録前に項目の確認画面が表示され，Web ユーザの許諾によりデータベースに form の値が格納される．表 3 中の装飾子

表 3　form 構成のための装飾子
Table 3　Modifiers for form structure.

装飾子名	値	機能	使用例
textarea	-	複数行入力フォーム	@{textarea}
password	-	パスワード入力	@{password}
radio	選択肢 1[|選択肢 2[|...]] ／ 選択肢 1=DB 挿入値 [|選択肢 2=DB 挿入値 [|...]] ／ 無し (装飾子 sql 使用時)	選択 (ラジオボタン)	@{radio= 'Yes=1|No=2|N/A=3'}
selectbox	選択肢 1[|選択肢 2[|...]] ／ 選択肢 1=DB 挿入値 [|選択肢 2=DB 挿入値 [|...]] ／ 無し (装飾子 sql 使用時)	選択 (セレクトボックス)	@{selectbox= ' 出席=1|欠席=2|未定=3'}
checkbox	選択肢 1[|選択肢 2[|...]] ／ 選択肢 1=DB 挿入値 [|選択肢 2=DB 挿入値 [|...]] ／ 無し (装飾子 sql 使用時)	選択 (チェックボックス)	@{checktbox= 'Cat=1|Dog=2'}
sql	SQL 文 (第一属性値は選択肢として， 第二属性値は DB 挿入値として使用． 第三属性値以降は無使用)	データベースの値を選択値と して使用 (radio，selectbox， checkbox と併用)	@{selectbox, sql='select title, id from movie'}
date	-	日付 (年/月/日)	@{date}
date1	-	日付 (年)	@{date1}
date2	-	日付 (月)	@{date2}
date3	-	日付 (日)	@{date3}
date4	-	日付 (年/月)	@{date4}
date5	-	日付 (月/日)	@{date5}
time	-	時刻 (時:分)	@{time}
url	-	URL	@{url}
email	-	メールアドレス	@{email}
number	-	数値	@{number}
alphabet	-	アルファベット	@{alphabet}
file	保存先パス	ファイル全般	@{file=' 保存先パス'}
image	保存先パス	画像ファイル	@{image=' 保存先パス'}
audio	保存先パス	音声ファイル	@{audio=' 保存先パス'}
video	保存先パス	映像ファイル	@{video=' 保存先パス'}
notnull	-	入力必須	@{notnull}
placeholder	文字列	プレースホルダテキスト	@{placeholder=' 感想を入力'}
min / max	数値	最小/最大 文字列長	@{max=100}
js / php / css	ファイル名 (カンマ区切りで複数指定)	処理に使用する JavaSctipt/PHP/CSS の指定	@{js=' ファイル名'}
value	値	DB 挿入値 (固定値)	@{value=' 値'}
hidden	-	隠し項目	@{hidden}
disabled	-	項目の無効化	@{disabled}
noinsert	-	DB 挿入は行わない	@{noinsert}
submit	-	form submit (テキスト)	@{submit}
submit_button	-	form submit (ボタン)	@{submit_button}
reset	-	form reset (テキスト)	@{button}
reset_button	-	form reset (ボタン)	@{reset_button}

が指定されていなかった場合，その属性は通常のテキスト入力項目となる．表中の装飾子 radio，checkbox の各選択項目は水平方向に表示される．この選択項目を垂直方向へ表示させる場合は，各装飾子の先頭の “v” を付加する．また，本機能では，データベースに格納されている値を form の選択肢として使用する機構（5.4 節）を表中の装飾子 sql として実装している．

5.　埋め込み型 SuperSQL：処理系

5.1　PHP 関数

ここでは，SuperSQL を PHP 上で動作させるために実装した PHP の関数の具体的な内容について説明する．また，これらの PHP 関数は，PHP Extention として実装することを意図して開発を行っている．本手法の PHP 関数部の処理の流れを図 4 に示す．

5.1.1　ssql_setConfig 関数

本関数は SuperSQL の設定（3.3 節）の読み込みを行う．引数には設定ファイル（3.3 節）のパスの指定もしくは，設定項目の直接指定（データベースドライバ，データベース名等）が可能である．本関数の内部では，引数で受け取った値に従って SuperSQL 実行の際の引数（3.4 節）へと変換する処理が行われる．本関数の引数を入力とした場合の処理アルゴリズムを下記に示す．戻り値は，SuperSQL の実行引数となる．

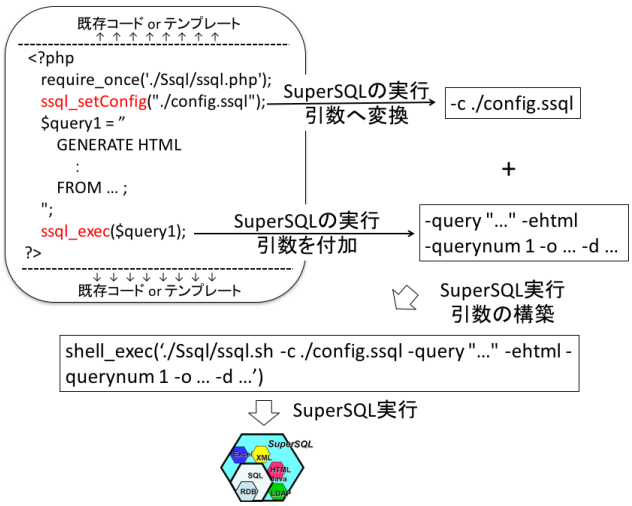

図 4　埋め込み型 SuperSQL の PHP 関数処理の流れ
Fig. 4　Processing flow of PHP functions of Embedded SuperSQL.

Algorithm 1

```
switch ssql_setConfig() の入力引数 do
    case 引数が NULL return Error
    case 引数が一つ return -c [第一引数の値]
    case OTHERS return -driver [第一引数の値] -db
                       [第二引数の値] -h [第三引数の値] -u
                       [第四引数の値] -p [第五引数の値]
```

5.1.2　ssql_exec 関数

本関数は SuperSQL の実行を行う．この関数を PHP の関数として実現することで，任意のタイミングでの SuperSQL

実行が可能となり，動的な Web コンテンツの埋め込みを実現した．本関数の実現に際して SuperSQL に新たに実装した実行引数（3.4 節）を以下に示す．

- -query ‘*[SuperSQL クエリ]*’
  指定したクエリを実行
  例：-query ‘GENERATE HTML *<TFE>* FROM $r_1, r_2, \ldots, r_n$ WHERE P”
- -ehtml
  埋め込み型 SuperSQL の実行
- -incremental
  埋め込み型 SuperSQL のデータ更新実行，もしくは簡易実行（5.6.2 項，5.6.3 項）
- -querynum *[ページ内の SuperSQL クエリの連番]*
  ページ内のクエリの連番の設定
- -ehtmlarg *[{値 1，値 2，...}]*
  受け渡す値の設定（5.6.2 項）

Standalone SuperSQL（GENERATE HTML）では主に実行引数 f を使用し，クエリファイル（3.2 節）を指定して SuperSQL の実行を行っていたが，提案手法ではクエリファイルの実行ではなくクエリを直接指定するため，実行引数 query の実装を行った．また，Standalone SuperSQL と提案手法の実行処理系の分岐を行うために実行引数 ehtml，incremental（5.6.2 項，5.6.3 項）を追加した．実行引数 querynum は，ページ内で複数の SuperSQL クエリが実行された際に，SuperSQL の処理系がそれぞれのクエリを判別できるようにページ内の SuperSQL クエリの ID の設定を行っている．この実行引数は 5.6 節の処理で利用する．

本関数の引数を入力とした場合の処理アルゴリズムを下記に示す．

Algorithm 2

```
1: if ssql_exec() の入力引数 != NULL then
2:     if ssql_setConfig() != Error then
3:         SuperSQL の実行 :
           shell_exec("./Ssql/ssql.sh "+ssql_setConfig()
           +" -query "+ 引数の値 +" -ehtml -querynum …
           -o … -d …")
4:     else
5:         Error
6:     end if
7: else
8:     Error
9: end if
```

本関数の内部では，ssql_setConfig 関数（5.1.1 項）によって変換された SuperSQL の実行引数に加え，実行引数 query，ehtml，incremental の追加を行い，SuperSQL 本体の実行を行うシェルスクリプトファイル（ssql.sh）を実行する処理を行っている．

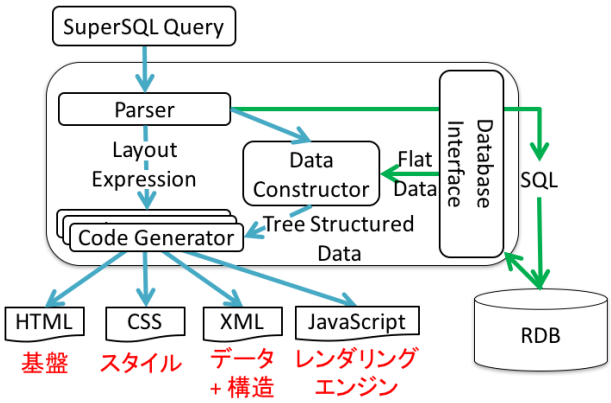

図 5　提案手法の実行の内部処理
Fig. 5　Internal Processing of Execution of Embedded SuperSQL.

5.2　PHP 埋め込み型 SuperSQL の実行の内部処理

5.1.2 項の関数によって実行される SuperSQL の内部処理について本節で説明する．実行の内部処理について図 5 に示す．

本処理は，以下の入出力をとる．

入力：SuperSQL クエリ
出力：HTML，CSS，XML，JavaScript ファイル

まず，入力された SuperSQL クエリを構文解析部で解析し，「SQL クエリ」と「出力する HTML の構造や装飾等の情報を持つレイアウト式」に分割する．次に，データベースインタフェース部を通して関係データベースに前出の SQL クエリの問合せを行い，結果をデータコンストラクタへ受け渡す．データコンストラクタは，受け取ったフラットなデータをツリー構造のデータに変換し，コード生成部へ受け渡す．最後にコード生成部では，受け渡された情報を基にして各出力ファイルの生成を行い，出力する．

これらの出力ファイルを用いてクライアントサイドで Web コンテンツの構築を行う．クライアントサイドで構築された Web コンテンツは最終的に，生成された基盤 HTML にアペンドし，既存コード内に埋め込まれる．

5.3　埋め込み型 SuperSQL のパラメータ受け渡し機構

5.3.1　plink，glink 関数と parameter 句

4.3 節で説明した plink，glink 関数をリンク元のページ，parameter 句をリンク先のページのクエリにそれぞれ使用していた場合，まず，Web 利用者がリンク元のページを開いた際（図 6 ①）に，サーバサイドで SuperSQL が実行（図 6 ②）され，ハイパーリンクが生成される（図 6 ③）．このとき，plink 関数を使用していた場合は POST 形式，glink 関数の場合は GET 形式の値の受け渡しリンクとなる．続いて，Web 閲覧者によってリンク元のページのハイパーリンクがクリックされ（図 6 ④），parameter 句を使用したクエリが埋め込んであるページ（図 6 parameter.php）

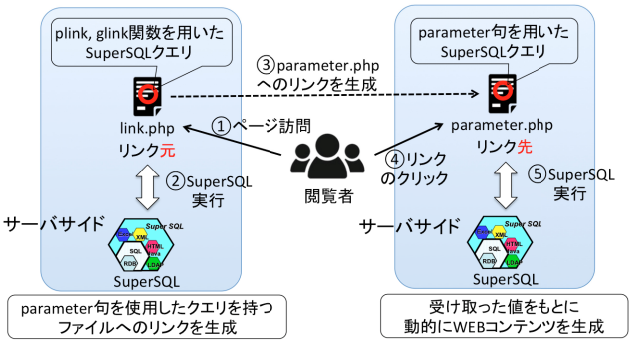

図 6　plink，glink 関数と parameter 句の処理
Fig. 6　Processing of plink function, glink function and parameter clause.

へのアクセスが発生した際にも，同様にサーバサイドで SuperSQL が実行（図 6 ⑤）され，リンク元のページから受け取った値を基にしてリンク先のコンテンツが動的生成される．

5.3.2　SuperSQL への値の受け渡し

図 6 の ⑤ で実行される SuperSQL の処理について説明する．⑤ では，リンク先のコンテンツを生成するために必要な値の受け渡しを行っている．受け渡しは，SuperSQL の新たな実行引数（3.4 節）として ehtmlarg とその処理を追加することで実現した．まず，Web 利用者によりリンク元のページのハイパーリンクが選択された際に，POST 形式，もしくは GET 形式で PHP 経由による値の送信が行われる．そしてリンク先のページで，受信した値を下記のように SuperSQL の実行引数に -ehtmlarg *{値 1，値 2，…}* といった形で指定して SuperSQL へ受け渡す．

Algorithm 3
```
1: if ssql_setConfig() != Error then
2:     SuperSQL の実行 :
       shell_exec("./Ssql/ssql.sh "+ssql_setConfig()
       +" -query "+ 引数の値 +" -ehtml -querynum …
       -o … -d … -ehtmlarg { 値 1, 値 2, …}")
3: else
4:     Error
5: end if
```

上記によって実行される SuperSQL のデータベースインタフェース部の処理の一部を以下に示す．

Algorithm 4
```
1: if 引数 ehtmlarg != NULL && parameter 句 != NULL then
2:     構文解析後，SQL クエリの WHERE 句に"parameter 句の
       属性名=ehtmlarg の値"を追加して問い合わせを実行
3: else
4:     通常の SQL 問い合わせを実行
5: end if
```

本処理では，parameter 句と実行引数 ehtmlarg により受け取った値を SQL クエリに条件式として付加して問合せを実行している．以下に例を示す．

```
PARAMETER e.id
GENERATE HTML
[e.name, e.salary]!
FROM employee e;
```

まず，上記の SuperSQL クエリが構造解析部での処理を経て，以下の SQL に変換される．

```
SELECT e.name, e.salary
FROM employee e;
```

その後，SQL 問合せを行う直前に parameter 句により与えられた属性と実行引数 ehtmlarg により与えられた値を基にして以下のように WHERE 句の補完が行われる．

```
SELECT e.name, e.salary
FROM employee e
WHERE e.id = parameter 句で受け取った値;
```

そして，上記の SQL を基にして問合せが行われる．

5.4　装飾子 form の処理機構

4.3 節で説明した提案手法の form 機能では，入力値や装飾子（3.2.3 項）の値として記述された項目のデータベースへの格納に加えて，データベースに格納されている値を form の選択肢として使用する機構を表 3 中の装飾子 sql として実装している．装飾子 sql の処理の一部を以下に示す．

Algorithm 5
```
1: if 装飾子 sql != NULL then
2:     if 右辺の SQL を実行 != Error then
3:         form の選択肢 = SQL の第一引数の結果
4:         if SQL の第二引数の結果 != NULL then
5:             データベースに格納する値 = SQL の第二引数の結果
6:         else
7:             データベースに格納する値 = SQL の第一引数の結果
8:         end if
9:     end if
10: end if
```

本処理は，ページ閲覧時に PHP が実行されるタイミングで，サーバサイドで，装飾子として指定された SQL 文によるデータベースからのデータの取得が行われる．そして取得した値の第 1 属性値を form の選択肢として，第 2 属性値をデータベース格納時の値として使用する．第 2 属性が未指定の場合，データベースには第 1 属性の値が格納される．また，第 3 属性以降が指定されていた場合，その値は使用されない．以下に例を示す．

```
GENERATE HTML {
  {"従業員氏名：", employee_id@{selectbox,
   sql='SELECT e.name, e.id FROM employee e'}}!
```

```
    {“コメント：”, comment@{textarea, notnull}}
  }@{form}
  FROM questionnaire;
```

上記の SuperSQL クエリは，構造解析部の処理を経て，コード生成部の処理へ入る．コード生成部では装飾子 form が指定されている <*T F E*>（3.2 節）内が form 機能の生成部として処理され，各指定装飾子に従って form が構成される．上記のクエリでは，questionnaire テーブルの employee_id 属性への挿入値として従業員氏名の選択肢（セレクトボックス），comment 属性への挿入値の入力欄として入力必須の複数行入力エリアを持ったコメント入力欄が form として生成される．employee_id 属性の装飾子として指定されている装飾子 @{selectbox, sql=’*SQL 文*’} は，構文解析により左辺と右辺の値に分解される．そして，装飾子 sql の右辺の値の SQL 文（SELECT e.name, e.id FROM employee e）によりデータベースからデータの取得が行われ，取得された値の第 1 属性 e.name の値が form の選択肢として，第 2 属性 e.id の値が form 登録時のデータベース格納値として使用される．

本処理では，Web ユーザによる form 登録時の値のバリデーション用スクリプトや，データベース格納処理のための SQL を含んだ PHP コードもあわせて生成される．

5.5　クライアントサイド処理機構

5.5.1　テンプレートの CSS との衝突の回避

提案手法を実現するにあたってあげられた問題点が，テンプレート等の既存コードに本手法を埋め込むことによって生成される Web コンテンツと，既存コードの衝突である．そこで，提案手法では，生成される Web コンテンツに対してユニークな ID の割当てを行い，既存コードとスタイルの干渉が起きないようにした．

生成された CSS は，提案手法の実行パス内の’jscss/ファイル名/ssqlResult+*ファイル内のクエリの ID（連番）*.css’として保存される．たとえば，index.html というファイル内の 1 つ目の SuperSQL クエリの SQL の CSS は，‘jscss/index/ssqlResult1.css’ として保存される．

5.5.2　クライアントサイドでの Web コンテンツの構築

提案手法では，Web コンテンツの構築を XML と JavaScript を用いてクライアントサイドで行う機構を実装した．SuperSQL レンダリングエンジンである JavaScript はすべてのクエリの実行で共通に使用される．処理の流れを図 **7** に示す．まず，任意のタイミングで実行された提案手法の SuperSQL によって XML（5.5.2.1），JavaScript（5.5.2.2），CSS 等のファイルが生成される．そして，クライアントサイドでレンダリングエンジンである JavaScript を実行し，生成されたファイルを用いて Web コンテンツを生成する．

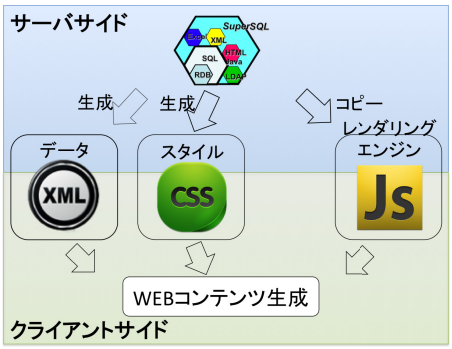

図 7　クライアントサイドの Web コンテンツ構築の流れ
Fig. 7　Flow of client-side Web contents construction.


```
1  <?xml version=‘‘1.0’’ encoding=‘‘UTF–8’’?>
2  <Grouper0 type=‘G2’ outType=‘div’ class=‘TFE10018’>
3    <Connector0 type=‘C2’ outType=‘div’ class=‘TFE10017’>
4      <Value outType=‘div’ class=‘TFE10000’>思い出のマーニー (2014)</TFE10000>
5      <Connector1 type=‘C1’ outType=‘div’ class=‘TFE10008’>
6        <Img class=‘TFE10001’ src=‘./img/movie23.jpg’></img>
7        <Connector2 type=‘C2’ outType=‘div’ class=‘TFE10007’>
8          <Value outType=‘div’ class=‘TFE10005’>作品紹介</TFE10005>
9          <Value outType=‘div’ class=‘TFE10006’>心を閉ざした少女杏奈は、
                    :
                が起きるようになるが、それは二人だけの秘密だった。</TFE10006>
10       </Connector2>
11     </Connector1>
12     <Value outType=‘div’ class=‘TFE10009’>レビュー</TFE10009>
13     <Grouper1 type=‘G2’ outType=‘div’ class=‘TFE10016’>
14       <Connector1 type=‘C1’ outType=‘div’ class=‘TFE10015’>
15         <Img class=‘TFE10010’ src=‘./img/2.png’></img>
16         <Value outType=‘div’ class=‘TFE10014’>話の流れが良くなかった、…
             </TFE10014>
17       </Connector1>
18       <Connector1 type=‘C1’ outType=‘div’ class=‘TFE10015’>
19         <Img class=‘TFE10010’ src=‘./img/4.png’></img>
20         <Value outType=‘div’ class=‘TFE10014’>難しかった</TFE10014>
21       </Connector1>
22     </Grouper1>
23   </Connector0>
24   <Connector0 type=‘C2’ outType=‘div’ class=‘TFE10017’>
25     ...
26   </Connector0>
27   ...
28 </Grouprer0>
```

生成されたXMLファイルの一部

図 8　生成される XML ファイルの例
Fig. 8　Example of generated XML file.

5.5.2.1　データと構造（XML）

図 7 の SuperSQL によって生成される XML について説明する．本 XML はデータと構造の役割を担っており，以下の情報を保持している．

- データベースから取得した値
- 結合子，反復子等の構造情報
- データレイアウトの出力形式（div タグ，table タグ）
- <*T F E*> の ID 情報（<*T F E*> の各属性には，CSS 指定等のための固有の ID が割り当てられている）

また，XML は入れ子の構造をとっている．図 **8** に提案手法で生成された Web コンテンツと生成された XML ファイルの一部を示す．

要素名は，結合子は Connector，反復子は Grouper，値

は Value，画像は Img としている．結合子と反復子の要素名には階層を示す数値を付加している．outType 属性で div タグ，table タグのどちらで出力するかを指定している．class 属性は class の指定を行っており，デフォルトでは $<TFE>$（3.2 節）の各属性に割り当てられている固有の ID 情報（TFE10000 から始まる連番）が設定されているが，クエリ内での装飾子@{class= クラス名} によってユーザ指定の class を設定することも可能である．type 属性では，クエリの結合子と反復子を保持している．

なお，本 XML ファイルは，提案手法が配置されているディレクトリに'GeneratedXML/ファイル名/ssqlResult+ファイル内のクエリの ID（連番）.xml' として生成される．

5.5.2.2　SuperSQL レンダリングエンジン（JavaScript）

図 7 の SuperSQL によって生成される JavaScript について説明する．本 JavaScript は，XML を解析し Web コンテンツを構築するレンダリングエンジンの役割を担っており，提案手法における実質的なコードジェネレータに相当する．本レンダリングエンジンの処理の一部を以下に示す．入力は 5.5.2.1 の XML，出力は Web コンテンツとなる．

Algorithm 6

```
1: if 入力 XML != NULL then
2:     str = ”
3:     for XML の一行目 to 最終行 do
4:         switch 現在の行のタグ do
5:             case C1, G1 開始タグ return str += 横結合開始タグ
6:             case C1, G1 終了タグ return str += 横結合終了タグ
7:             case C2, G2 開始タグ return str += 横結合開始タグ
8:             case C2, G2 終了タグ return str += 縦結合終了タグ
9:             case Img return str += img タグ
10:            case Value return str += 値
11:                :
12:    end for
13:    return str
14: else
15:    return Error
16: end if
```

提案手法を利用しているページ内のすべての SuperSQL クエリの実行が完了したのち，クライアントサイドで本エンジンを用いて XML ファイルの読み込み，解析，Web コンテンツの構築を行う処理を，ページ内のクエリの数だけ再帰的に行う．

5.6　キャッシュを利用した実行の分岐機構

提案手法では，ページ閲覧ユーザがページを閲覧するたびに SuperSQL がサーバサイドで実行される．そのため，提案手法を利用したページに対する閲覧回数が多ければ多いほど，SuperSQL を実行するサーバに負担がかかってしまう．また，ページ内のクエリが変更されていない場合，Web コンテンツを載せる HTML コードや CSS，JavaScript 等に変化はないにもかかわらず，毎回の実行でこれらを再生成，再コピーしてしまうのは，冗長であると考えられる．加えて，ページ内のクエリで使用されているデータベースに変更がなかった場合，XML の再生成も冗長になってしまう．

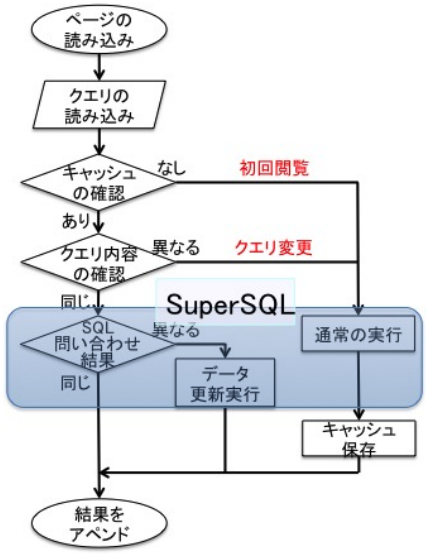

図 9　キャッシュを利用した実行の分岐のフローチャート
Fig. 9　Flow chart of execution using cache.

そこで，提案手法ではページ内のクエリに変更がなかった場合は，Web コンテンツを載せる HTML コードや CSS，JavaScript 等を前回の SuperSQL の実行で生成，コピーされたものを再利用し，XML の生成，すなわちデータの更新のみを行うデータ更新実行と，クエリに変更がなく，さらにデータベースにも変更がなかった場合は XML も前回のものを再利用する機構を実装した．これらの提案手法の実行方法を以下に示す．

- 通常の埋め込み実行
  - 不特定のユーザによる初回ページ訪問時に実行される．
  - 生成物：構築された Web コンテンツを載せる基盤 HTML コード，XML，CSS，JavaScript（コピー）
- データ更新実行
  - 2 回目以降の実行でクエリに変更がなく，データベースに更新があった場合に実行される．
  - 生成物：XML
- 簡易実行
  - 2 回目以降の実行でクエリに変更がなく，データベースに更新がなかった場合に実行される．
  - 生成物：なし

この機構が実現することで，SuperSQL が実行されるサーバの負荷を軽減することができる．上記 3 つの実行方法の分岐には，キャッシュを利用した．キャッシュを利用した実行の分岐のフローチャートを図 9 に示す．

5.6.1　PHP を利用したキャッシュ

提案手法へのキャッシュシステムの導入には，PHP のキャッシュライブラリである PHP PEAR の Cache_Lite [31] を使用している．このライブラリを使用した理由は，軽量でありながら詳細な設定が可能なためである．提案手法でこのライブラリを使用するうえでの設定は以下のようになっている．

'cacheDir' でキャッシュを保存するディレクトリを指定している．提案手法では，提案手法の実行パス内の tmp ディレクトリに設定している．なお，ここで指定するディレクトリには書き込み権限を与える必要がある．

```
1: /**
2:  * キャッシュオプション設定
3:  */
4: $cacheOptions = array(
5:     ' cacheDir ' =>' ./ Ssql/tmp/ ' ,
6:     ' caching ' =>' true ' ,
7:     ' automaticSerial ization ' =>' true ' ,
8:     ' lifeTime ' =>' 86400 '
9: );
```

'caching' ではキャッシュ機能を有効にするかを設定している．'automaticSerialization' では配列の保存可能にするかを設定することができ，提案手法では複数の値を配列としてセットで保存しているため，有効としている．'lifeTime' では保存したキャッシュの生存時間を指定することができる．提案手法では，この生存時間を 24 時間に設定している．ゆえに，1 日に 1 回は必ずキャッシュがリセットされ，通常の埋め込み実行が実行されることとなる．

なお，提案手法でキャッシュする値は，クエリ，構築された Web コンテンツを載せる基盤 HTML コードの 2 つである．これらは配列形式でそれぞれのキャッシュを識別する ID ごとに保持している．キャッシュの ID は' ファイル名 +Query+ ファイル内のクエリの ID（連番）' となっている．たとえば，index.html というファイル内で 2 つの SuperSQL クエリがあった場合，1 つ目のクエリのキャッシュ ID は，indexQuery1 となり，2 つ目のクエリのキャッシュ ID は，indexQuery2 となる．キャッシュで保存するクエリはデータベースの情報が記述されているため，MD5 を使用して 128 ビットのハッシュ値に変換してから保存している．

提案手法にこの処理機構を実装するにあたり，通常の埋め込み実行の SuperSQL の処理系にも若干の修正を加えた．**図 10** に埋め込み実行の埋め込み型 SuperSQL の処理の流れを示す．変更点は，SuperSQL のクエリが通常の SQL に変換され，関係データベースに問合せを行うときである．ここで，データ更新実行，簡易実行の際に検索結果を比較す

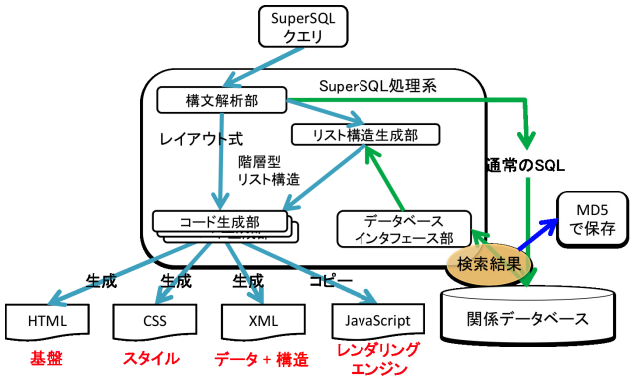

図 10　通常の埋め込み実行の埋め込み型 SuperSQL の内部処理
Fig. 10　Internal processing of Embedded SuperSQL execution.

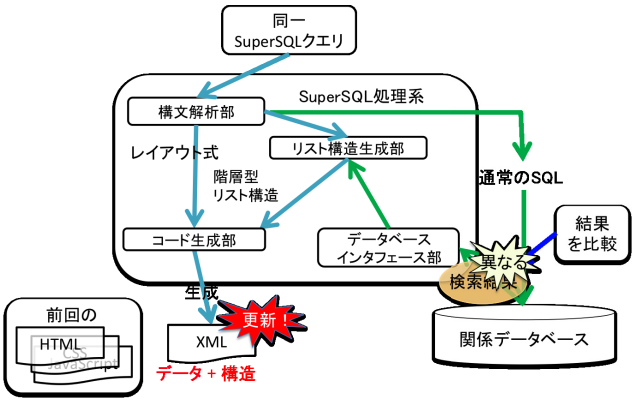

図 11　データ更新実行の埋め込み型 SuperSQL の内部処理
Fig. 11　Internal processing of Embedded SuperSQL data update execution.

るため，検索結果を MD5 でハッシュ値に変換してテキストファイルにして提案手法の実行パス内の 'sqlResult/ファイル名/ssqlResult+ ファイル内のクエリの ID（連番）.txt' として保存している．たとえば，index.html というファイル内の 1 つ目の SuperSQL クエリの SQL の問合せ結果は，'sqlResult/index/ssqlResult1.txt' として保存される．MD5 を用いている理由は，検索結果の量が多く，サイズが大きくなってしまっても MD5 であれば固定長の非常に小さいサイズで変換可能なためである．

また，本処理工程では検索結果が同じものかどうかの判断のみできればよいため，比較するときも元に戻さずにハッシュ値どうしを比較することができるのも利点である．さらに，MD5 では同じ入力値からは必ず同じ値が得られる．一方，少しでも異なる入力値であればまったく違う値が得られるという点もこの機構には適しているといえる．

5.6.2　データ更新実行

有効なキャッシュがあり，クエリに変更がなく，データベースに更新があった場合に実行されるデータ更新実行について説明する．データ更新実行の埋め込み型 SuperSQL の処理の流れを**図 11** に示す．データ更新実行では，SuperSQL のクエリ解析部の処理が終わり，SuperSQL クエリから通

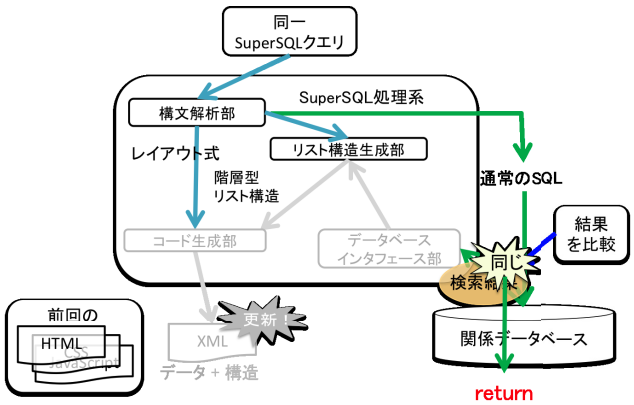

図 12　簡易実行の埋め込み型 SuperSQL の内部処理
Fig. 12　Internal processing of Embedded SuperSQL simple execution.

常の SQL に変換されてデータベースに問合せを行った際に，検索結果を MD5 ハッシュ値に変換した値と保存済みのクエリの検索結果（MD5 ハッシュ値）と比較し，一致していなかった場合，すなわちデータベースに更新があった場合は，構築された Web コンテンツを載せる基盤 HTML コード，CSS の生成，JavaScript のコピーを行う必要がなく，前回の実行で生成，コピーされたものを使用することができるため，コード生成部でデータと構造の役割を担う XML ファイルのみを再生成する処理の流れとなっている．

5.6.3　簡易実行

有効なキャッシュがあり，クエリに変更がなく，データベースに更新がなかった場合に実行される簡易実行について説明する．簡易実行の埋め込み型 SuperSQL の処理の流れを図 **12** に示す．データ更新実行では，SuperSQL のクエリ解析部の処理が終わり，SuperSQL クエリから通常の SQL に変換されてデータベースに問合せを行った際に，検索結果を MD5 ハッシュ値に変換した値と保存済みのクエリの検索結果（MD5 ハッシュ値）と比較し，一致していた場合，すなわちデータベースに変更がなかった場合は，XML ファイルも前回生成したものを使用することができるため，XML も再生せずに処理を終える流れとなっている．

6.　埋め込み型 SuperSQL：用例

6.1　概要

本章では，提案手法の用例として，映画のレビュー閲覧・投稿サイトを埋め込み型 SuperSQL を用いて作成する例を示す．本章の各例を PHP のみを用いて作成した場合，より多くのコードを書く必要が生じ，また，レイアウトの変更が発生した場合，内容によってはその変更は容易ではない．本提案手法が具体的にどのように PHP 開発者の負荷を軽減させることができるのかを示す目的で本章を執筆した．

本章の用例では以下の簡略化した 4 テーブルを使用する．各テーブルには，映画，映画ジャンル，製作会社，レ

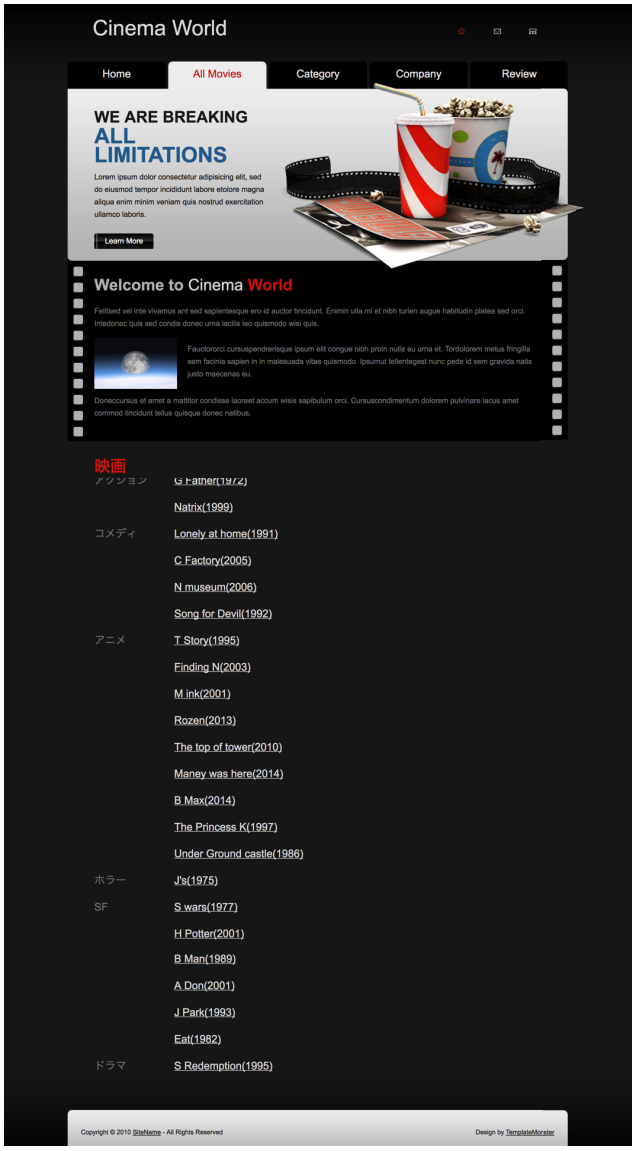

図 13　ジャンル別の映画名一覧ページ（movie_list.html）
Fig. 13　Movie title list page (movie_list.html).

ビューの情報が格納されている．

- movie (id, title, genre, company, year, image, abst)
- genre (id, name)
- company (id, name)
- review (id, m_id, comment, rank)

図 **13** のジャンル別の映画名一覧ページを作成している．本 HTML では，テンプレートの 65 行目に HTML コード，66〜76 行目の PHP 内に提案手法の PHP 関数を用いた SuperSQL クエリの埋め込み（合計 9 行）を行っている．具体的な処理としては，67 行目で埋め込み型 SuperSQL 用ファイルの読み込み，68 行目の ssql_setConfig() 関数でデータベースの情報が書かれた SuperSQL 設定ファイル（./config.ssql）の読み込み，75 行目の ssql_exec() 関数で引数として渡された変数（$query1）に格納されている SuperSQL クエリの実行を行っている．70〜73 行目の SuperSQL クエリでは，71 行目の <*T F E*> 指定において，

movie_list.html の一部

```
 1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/
       DTD/xhtml1-strict.dtd">
 2 <html xmlns="http://www.w3.org/1999/xhtml">
 3 <head>
 4 <title>Cinema World</title>
 5 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
 6 <link href="css/style.css" rel="stylesheet" type="text/css" />
 7 <script src="js/jquery-1.4.2.min.js" type="text/javascript"></script>
 8 <script src="js/cufon-yui.js" type="text/javascript"></script>
 9 <script src="js/cufon-replace.js" type="text/javascript"></script>
10 <script src="js/Gill_Sans_400.font.js" type="text/javascript"></script>
11 <script src="js/script.js" type="text/javascript"></script>
12 <!--[if lt IE 7]>
13 <script type="text/javascript" src="js/ie-png.js"></script>
14 <script type="text/javascript">ie_png.fix('.png, .link1 span, .link1');</script>
15 <link href="css/ie6.css" rel="stylesheet" type="text/css" />
16 <![endif]-->
17 </head>
18 <body id="page1">
19 <!-- START PAGE SOURCE -->
20 <div class="tail-top">
                :
63          <div class="content">
64
65            <h3> <span>映画</span></h3>
66            <?php
67              require_once('./Ssql/ssql.php');
68              ssql_setConfig("./config.ssql");
69              $query1 = "
70                GENERATE HTML
71                [g.name@{border=0, width=150, font-size=20}, [{plink(m.title || '('||m.
                      year || ')'@{height=50, width=400, font-size=20, border=0}, './
                      movie_detail.html', m.id)}]!]!
72                FROM movie m, company c, genre g
73                WHERE m.company=c.id AND m.genre=g.id
74              ";
75              ssql_exec($query1);
76            ?>
77
78          </div>
                :
93 </div>
94 <script type="text/javascript"> Cufon.now(); </script>
95 <!-- END PAGE SOURCE -->
96 </body>
97 </html>
```

また，提案手法の埋め込みには Web から入手したフリーの 84 行のテンプレート [32] を使用した．

6.2　用例

この節では，埋め込み型 SuperSQL を用いて作成した映画のレビュー閲覧・投稿サイトの 3 つの Web ページを例として取り上げる．

6.2.1　plink() の用例：movie_list.html

下記に示す 97 行の HTML（movie_list.html）により，映画のジャンル名（g.name）ごとにグルーピングし，plink() 関数を用いてハイパーリンクを選択することにより，映画タイトル（m.title）と放映年（m.year）を含む POST 形式のハイパーリンクを作成し，リンクが押された場合の飛び先を ./movie_detai.html，リンク先へ受け渡す値の属性を m.id として指定している．この受け渡す属性の情報に従ってリンク先のページ（movie_detail.html，6.2.2 項）が動的に生成され表示される．

また，前章で説明したとおり，本ページに埋め込まれている SuperSQL クエリによりデータベースから取得したデータは XML として保管され，クエリ，データベースともに変更がなかった場合は，XML の再発行は行われず，前回 XML が再利用される（指定されている一定間隔を経過し，キャッシュがリセットされている場合を除く）．

6.2.2　parameter 句の用例：movie_detail.html

下記に示す 29 行のコードを前項と同様にテンプレートの 65 行目に埋め込むことにより，**図 14** のレビューを含んだ映画情報ページを作成している．このページでは，2 つの SuperSQL クエリにより映画の詳細，およびレビュー情報の表示を行っている．6〜14 行目の SuperSQL クエリでは，6 行目の parameter 句でリンク元のページから受け取る値の属性（e.id）が指定され，その情報に従って本ページは動的に生成される．クエリの 8 行目では映画の画像（image），9 行目で映画タイトル（m.title）と放映年（m.year）の表示を行っている．10 行目では映画製作会社の名前（c.name）を plink() 関数を用いて表示させ，その名前を新たなリンク先（company_movie.html）へのハイパーリンクとしてい

movie_detail.html の一部

```
1  <h3><span>映画</span></h3>
2  <?php
3    require_once('./Ssql/ssql.php');
4    ssql_setConfig("./config.ssql");
5    $query1 = "
6      PARAMETER m.id
7      GENERATE HTML
8      [image(image, './img'),
9        {{m.title@{width=250, border=0, font-size=20, height=60}, '('||m.year||')'@{
             border=0}!
10       {plink(c.name@{width=220, border=0}, 'company_movie.html', c.id), plink(g.name@
             {width=130, border=0}, 'genre_movie.html', g.id)}}!
11       '作品紹介'@{border=0, height=30}!
12       m.abst@{width=360, height=210, border=0}} ]!
13     FROM movie m, company c, genre g
14     WHERE c.id = m.company AND g.id = m.genre
15   ";
16   ssql_exec($query1);
17 ?>
18
19 <h3><span>レビュー</span></h3>
20 <?php
21   $query2 = "
22     PARAMETER m.id
23     GENERATE HTML
24     [image(rank || '.png', './img'), r.comment@{border=0, width=500}]!
25     FROM movie m, review r
26     WHERE r.m_id=m.id
27   ";
28   ssql_exec($query2);
29 ?>
```

図 14　レビューを含んだ映画情報ページ（movie_detail.html）
Fig. 14　Movie information page (movie_detail.html).

図 15　映画レビュー投稿ページ（movie_review.html）
Fig. 15　Movie review contribution page (movie_review.html).

る．そして，11 行目では記述されたた文字列リテラルを，12 行目では映画の概要（m.abst）をそれぞれ表示させている．22〜26 行目の SuperSQL クエリでも同様に，22 行目の parameter 句で受け取る値の属性（e.id）を指定し，その情報に従って，24 行目で指定されている映画のレビュー評価（rank）とコメント（r.comment）を動的に生成して表示させている．

また，キャッシュを利用した実行の分岐等に関しては前項と同様である．

6.2.3　装飾子 form の用例：movie_review.html

下記に示す 17 行のコードを 6.2.1 項と同様にテンプレートの 65 行目に埋め込むことにより，**図 15** の映画レビュー投稿ページを作成している．このページでは，1 つの SuperSQL クエリにより映画のレビュー投稿 form の生成を行っている．6〜14 行目の SuperSQL クエリでは，7〜12 行目の装飾子 form で囲われた $<TFE>$ の情報に基づいて form が構成される．8 行目では装飾子 selectbox と sql の情報に基づいてデータベースから取得した映画タイトル（title）を form のセレクトボックスとして表示，9 行目では装飾子 vradio により右辺に指定されて項目をラジオボタンの項目として垂直方向へ表示，10 行目では装飾子 textarea と notnull により複数行入力の入力必須のテキストエリアを表示している．これらの 3 項目の選択値・入力値は，Web ユーザの許諾時にそれぞれ review テーブルの m_id，rank，comment 属性へ格納される．また，12 行目では，装飾子 submit_button により登録ボタン，装飾子 reset によりキャンセル処理を行う選択可能なテキストをそれぞれ生成している．

7.　実験・評価

7.1　概要

7.1.1　実験・評価項目

本章では，本手法の有用性を評価するために以下の評価実験を行った．

- Web で入手可能な様々なテンプレートに対する SuperSQL の埋め込み（7.2 節）

movie_review.html の一部

```
1  <h4><span>映画レビュー</span></h4>
2  <?php
3    require_once('./Ssql/ssql.php');
4    ssql_setConfig("./config.ssql");
5    $query1 = "
6      GENERATE HTML {
7        {
8          {"映画:"@{color='white', width='90px'}, m_id@{selectbox, sql='select title, id
               from movie', width='300px'}}!
9          {"評価:"@{color='white', width='90px'}, rank@{vradio='★☆☆☆☆=1|★★☆☆☆=2
               |★★★☆☆=3|★★★★☆=4|★★★★★=5', align='left', padding='5px'}}!
10         {"感想:"@{color='white', width='90px'}, comment@{textarea, notnull, width='300
               px', height='6em'}}
11       }@{table, border='1'}!
12       {"登録"@{submit_button, width='100px'}, "キャンセル"@{reset, font-size='13px',
               color='gray'}}
13     }@{form, font-size='17px'}
14     FROM review
15   ";
16   ssql_exec($query1);
17 ?>
```

- 各実行処理における SuperSQL の処理時間の評価
  - 通常の埋め込み実行（7.3.1 項）
  - データ更新実行（7.3.2 項）
  - 簡易実行（7.3.3 項）

7.1.2　実験環境

提案手法の評価実験環境は，CentOS 7.1，4 CPU，メモリ 32 GB，データベースは PostgreSQL9.4，PHP は PHP 5.4.16，ブラウザは FireFox 43.0.4 を使用した．

7.2　Web で入手可能な様々なテンプレートに対する SuperSQL の埋め込み

7.2.1　評価方法

提案手法の有用性を評価するために，Web 上で公開されている様々な HTML テンプレートに対して，提案システムを用いて，Web コンテンツの埋め込みを行い，実際にどの程度埋め込みを実現することができたかの実験を行った．実験には，以下の計 145 件のテンプレートを使用した．

- シンプルテンプレート（55 件）
- フレームワークを使用したテンプレート（90 件）
  - Bootstrap（14 件）
  - Material Design Lite（5 件）
  - Pure（8 件）
  - Foundation（24 件）
  - SkyBlue CSS（5 件）
  - HTML5 Boileropate（3 件）
  - UIkit（6 件）
  - その他フレームワーク（25 件）

7.2.2　評価結果

それぞれのテンプレートに対して提案手法を用いて SuperSQL クエリの埋め込みを行い，実際にどの程度既存コードに対して Web コンテンツを埋め込むことができたかを**表 4** に示す．「埋め込み」は，SuperSQL によるスタイルの指定を行わないデフォルトでの Web コンテンツの埋め込み，すなわち装飾子を使用せずに提案手法を使用した結果を示している．「CSS」は，SuperSQL で <*TFE*>（属性等）にスタイルの指定，すなわち装飾子を使用して提案手法を使用した結果を示している．最後に「JavaScript」は，ページネーション等の JavaScript を使用した Web コンテンツを生成するクエリの埋め込みを行った結果を示している．表 4 内の数値は，正常に提案手法が動作したテンプレート数の割合である．この結果，各埋め込みにおいて，SuperSQL によって生成した Web コンテンツをすべてのテンプレートの任意の場所に埋め込むことができることを確認した．

表 4　テンプレートに対する SuperSQL の埋め込み結果
Table 4　Results of Embedded SuperSQL towards HTML templates.

	埋め込み	CSS	JavaScript
Bootstrap（14件）	100%	100%	100%
Material Design Lite（5件）	100%	100%	100%
Pure（8 件）	100%	100%	100%
Foundation（24件）	100%	100%	100%
SkyBlue CSS（5件）	100%	100%	100%
HTML5 Boileropate（3件）	100%	100%	100%
UIkit（6件）	100%	100%	100%
その他フレームワーク（25件）	100%	100%	100%
シンプルテンプレート（55件）	100%	100%	100%
計145件	100%	100%	100%

7.3　コード量の比較

図 13, 図 14, 図 15 を既存手法（PHP, JavaScript, XML, CSS，HTML）を用いて作成した場合のコード量の比較を**表 5** に示す．コード量の列は，既存手法に対して提案手法では何割のコード量で実現することができたかを示してい

表 5　既存手法と提案手法のコード量の比較
Table 5　Comparison of code amounts.

	内容	既存手法	提案手法	コード量 (提案手法/既存手法)
1	ジャンル別の映画名一覧ページ（図13）	26行	9行	35%
2	レビューを含んだ映画情報ページ（図14）	69行	22行	32%
3	映画レビュー投稿ページ（図15）	97行	14行	14%

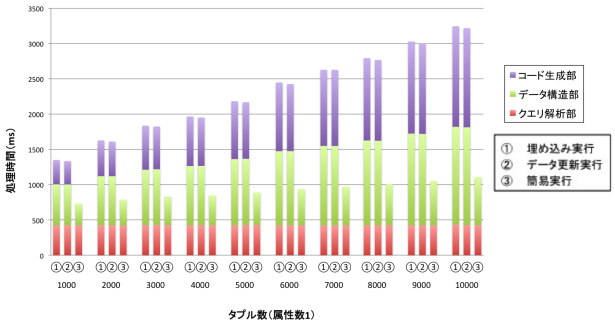

図 16　タプル数別処理時間の比較
Fig. 16　Comparison of processing time per number of tuples.

る．これらから，それぞれ既存手法と比較して 3 割以下のコード量で Web コンテンツ生成の実現ができたことが分かる．特に 3 の form を用いるページでは大幅なコード量の削減を確認した．

7.4　各実行処理における処理時間の評価

提案手法では，ページ閲覧ユーザがページを閲覧するごとに SuperSQL の実行が行われる．したがって，サーバへかかる負荷が大きくなってしまうことが考えられる．そこで本節では，提案手法で実装したキャッシュを利用した実行の分岐，XML のみの更新を行う実行と XML の更新も行わない実行によって，どの程度 SuperSQL の処理時間を削減することが可能かを使用するデータベースのタプル数，クエリ内の属性数，クエリ内のグルーピング数ごとに評価する．

7.4.1　タプル数別処理速度の比較

タプル数別処理速度の比較に用いた SuperSQL クエリは，タプル数の変化で各実行の処理時間にどのように影響するかを見るために，属性数は 1 つに固定している．そして，データベースのタプル数を 1000 タプルから 1000 タプルずつ 10000 タプルまで増加させ，処理時間の変化を評価する．

タプル数別に処理速度を比較した結果を図 16 に示す．

まず，キャッシュがあるときに XML ファイルのみの生成を行うデータ更新実行については，Web コンテンツを載せる基盤 HTML コード，CSS の生成と JavaScript のコピーを行わない分，キャッシュのない初回ページ訪問時に行われる埋め込み実行に比べて若干ではあるが，処理速度に改善が見られることが分かる．次に，キャッシュがあり，データベースの更新が行われていなかった場合に実行される簡易実行では，SuperSQL のデータ構造部の SQL への問合せが終了した時点で処理を終えているため，データ構造部の後半部とコード生成部の処理をすべて削減することができ，大幅な処理時間の削減が行えていることが分かる．タプル数が増えるにつれてその削減率は大きくなっていることが分かる．

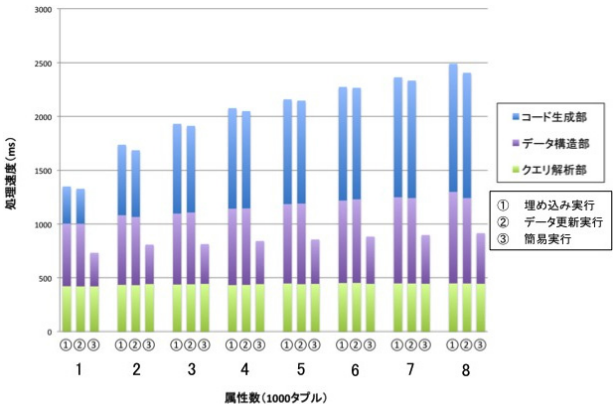

図 17　属性数別処理時間の比較
Fig. 17　Comparison of processing time per number of attributes.

7.4.2　属性数別処理時間の比較

属性数別処理時間の比較に用いた SuperSQL クエリは，クエリに使用される属性数の変化で各実行時間にどのように影響するかを見るために，タプル数を 1000 タプルに固定し，クエリに使用する属性数を 1 つから 8 つまで増加させ，処理時間の変化を評価する．

SuperSQL クエリ内のタプル数別に処理速度を比較した結果を図 17 に示す．7.4.1 項のタプル数別と同様に，データ更新実行では，属性数の変化に関係なく平均して約 20 ms の処理時間の改善が見られた．簡易実行でも，他の 2 つの実行と比べて大幅な処理時間の削減を確認した．

7.4.3　グルーピング数別処理時間の比較

グルーピング数別処理時間の比較に用いた SuperSQL クエリは，グルーピング数の変化で各実行時間にどのように影響するかを見るために，タプル数を 1000 タプル，属性数を 3 つに固定し，クエリに使用する入れ子のグルーピング数を 0 つから 2 つまで増加させ，処理時間の変化を評価する．

SuperSQL クエリ内のグルーピング数別に処理速度を比較した結果を図 18 に示す．7.4.1 項のタプル数別と同様に，データ更新実行では，グルーピング数の変化に関係なく平均して約 20 ms の処理時間の改善が見られた．簡易実行でもほか 2 つの実行と比べて，大きく処理時間の削減ができることが確認できたが，処理時間の削減率はグルーピング数が増えるに従って，低くなっていることが分かる．SuperSQL では，グルーピング数が増えると，生成する Web コンテンツのレイアウト構造が入れ子構造になり，複雑になるため，データ構造部での処理が増え，逆にグルーピングによる値の集約が大きいほどコード生成部の処理時間が大幅に削減される．簡易実行では，そのデータ構造を構築する前に処理を終えているため，グルーピングの影響を受けないため，このような結果になったと考えられる．

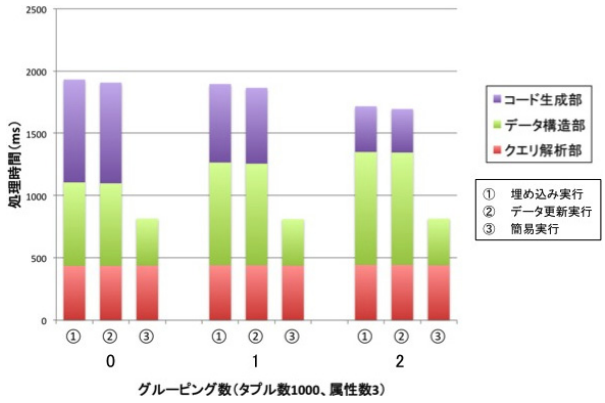

図 18　グルーピング数別処理時間の比較

Fig. 18　Comparison of processing time per number of groupings.

7.4.4　各実行処理における評価のまとめ

ここでは，キャッシュを利用した実行の有用性について述べる．XML の更新のみを行うデータ更新実行は，タプル数，属性数，クエリの構造に関係なく，平均して約 20 ms の処理時間の削減を行うことができた．Standalone SuperSQL では，クエリ内で大量の装飾子を使用しても生成される CSS は生成される XML ファイルと比べて非常に小さいため処理時間の削減率は小さくなっているが，将来的に，SuperSQL によってよりリッチな Web コンテンツの生成が可能になり，生成する CSS の容量が大きくなると，削減率はより大きくなることが期待できる．

データベースの更新がない場合に行われる簡易実行は，クエリの構造には左右されずに，データベースのタプル数，クエリに使用されている属性数が増えるにつれて処理時間を大きく削減することができ，簡易実行の有用性を示した．

8.　おわりに

8.1　まとめ

本研究では，データベースを利用した動的 Web サイトの開発を支援する埋め込み型 SuperSQL，およびその各処理機構を提案した．クライアントサイド処理機構では，データベースに格納されたデータを XML 形式で受け渡し，新たに JavaScript で実装した SuperSQL レンダリングエンジンを用いてその XML を解析し Web コンテンツを構築する仕組みを実現した．また，キャッシュを利用した実行の分岐機構では，データの更新のみを行うデータ更新実行と，クエリに変更がなくさらにデータベースにも変更がなかった場合は XML を再利用する仕組みを実装し，実行におけるサーバ負荷の軽減を実現した．本手法の提案機構は，PHP 以外の Java 等の他の Web 開発フレームワークにも適用可能である．

評価では，実際に Web で公開されている 145 件のテンプレートに対し，提案手法を利用した結果，(1) デフォルトスタイルの Web コンテンツの埋め込み：100%，(2) スタイル指定（CSS）した Web コンテンツの埋め込み：100%，(3) jQuery を使用した Web コンテンツの埋め込み：100%の埋め込みの実現を達成することができ，提案手法の有用性を示した．また，各実行処理における SuperSQL の処理時間の評価では，SuperSQL を実行するサーバにかかる負荷を軽減する機構の評価を行い，データ更新実行では，平均して 20 ms の処理時間の削減，簡易実行では，データベースのタプル数やクエリに使用している属性数が増えるほど大きく処理時間を削減できたことを示し，その有用性を示した．

8.2　今後の課題

提案手法では，1 章に示した条件 1)〜3) に即した Web 開発において，4.3 節で述べられている機能を利用することにより様々な Web ページの作成が可能となる．また，対象に則した Web 開発において，PHP や Ruby on Rails 等のサーバサイドプログラミング言語と比較して，図 1 に示すように少ないコード量（コスト）による処理の記述が可能である．しかしながら，現状，複雑な処理の記述には対応できていない．たとえば，リッチな Web クライアントを対象とした場合，基本仕様に沿って構築されたものは提案手法を適用可能であるが，複雑な構造をともなって構築されたものについては提案手法の拡張が必要である．その点は今後の課題である．

また，本稿では，公開されている動的 Web サイトの約 8 割が PHP を利用して作成されているという事実を考慮して PHP への埋め込み対応を行ったが，前述のとおり，本手法の提案機構は他の Web 開発フレームワークにも適用可能である．そこで今後は他の Web フレームワークへの対応を行う．また，SuperSQL のマルチ媒体出力という特性を活かした Web 開発言語以外の言語等への埋め込みも検討している．

参考文献

[1] W3Techs - World Wide Web Technology Surveys. 入手先 ⟨http://w3techs.com/⟩（参照 2017-07-01）.
[2] Toyama, M.: Supersql: an extended sql for database publishing and presentation, *Proc. ACM SIGMOD international conference on Management of data*, Vol.27, pp.584–586 (1998).
[3] 遠山元道：ターゲットリストの拡張によるデータベース出版と概視の実現，情報処理学会研究報告，データベース・システム研究会報告，Vol.93, No.65, pp.243–252 (1993).
[4] SuperSQL, 入手先 ⟨http://ssql.db.ics.keio.ac.jp⟩（参照 2017-07-01）.
[5] PHP, 入手先 ⟨https://secure.php.net/⟩（参照 2017-07-01）.
[6] ASP.NET, 入手先 ⟨http://www.asp.net/⟩（参照 2017-07-01）.
[7] Java, 入手先 ⟨https://www.java.com/ja/⟩（参照 2017-07-01）.
[8] static files, 入手先 ⟨https://docs.djangoproject.com/ja/1.9/howto/static-files/⟩（参照 2017-07-01）.

[9] ColdFusion，入手先 ⟨http://www.adobe.com/jp/products/coldfusion-family.html⟩（参照 2017-07-01）．
[10] JavaScript，入手先 ⟨https://developer.mozilla.org/ja/docs/Web/JavaScript⟩（参照 2017-07-01）．
[11] jQuery, 入手先 ⟨http://jquery.com/⟩（参照 2017-07-01）．
[12] Rode, J., Bhardwaj, Y., Pérez-Quiñones, A.M., Rosson, B.M. and Howarth, J.: As easy as "click ": End-user web engineering, *Proc. Web Engineering*, pp.478–488 (2005).
[13] Marriott, K., Moulder, P. and Hurst, N.: Html automatic table layout, *ACM Trans. Web*, Vol.7, No.1, pp.4:1–4:27 (2013).
[14] Samimi, H., Schäfer, M., Artzi, S., Millstein, T., Tip, F. and Hendren, L.: Automated repair of html generation errors in php applications using string constraint solving, *Proc. 34th International Conference on Software Engineering*, pp.277–287 (2012).
[15] Ducasse, D., Lienhard, A. and Renggli, L.: Seaside: A Flexible Environment for Building Dynamic Web Applications, *IEEE Software*, Vol.24, No.5, pp.56–63 (2007).
[16] Udoguchi, S., Iijima, T. and Toyama, M.: Application of SuperSQL query language for the migration from a relational to an object-oriented database, *Proc. International Database Engineering and Applications Symposium*, pp.207–216 (2000).
[17] 赤堀正剛, 有澤達也, 遠山元道：SuperSQL による関係データベースと XML データの統合利用，情報処理学会論文誌：データベース，Vol.42, No.SIG08(TOD10), pp.66–95 (2001).
[18] 笹田麻衣子，遠山元道：SuperSQL の時間連結子による動的プレゼンテーション生成，情報処理学会論文誌：データベース，Vol.46, No.SIG13(TOD27), pp.65–77 (2005).
[19] 有澤達也，石川恭子，遠山元道：SuperSQL の INVOKE 処理における中間データのキャッシュ，日本データベース学会 Letters，Vol.3, No.1, pp.33–36 (2004).
[20] 石川恭子，有澤達也，遠山元道：データ集約型 Web サイトにおける静的生成コンテンツの部分更新，情報処理学会論文誌：データベース，Vol.46, No.SIG13(TOD27), pp.1–15 (2005).
[21] Goto, K., Koshijima, R. and Toyama, M.: Generating Desktop and Mobile Web Pages from a Single SuperSQL Query, *Proc. 19th International Database Engineering & Applications Symposium*, pp.222–223 (2015).
[22] Kim, T-S., Shin, S-G. and Toyama, M.: Efficient Media Publication Using SuperSQL Processor, *Journal of Korea Society for Simulation*, Vol.15, No.1, pp.59–67 (2006).
[23] Shin, S-G., Kim, T-S. and Toyama, M.: Integrated Methods of Various Media Generators in The SuperSQL Query Process System, *Journal of Korea Multimedia Society*, Vol.9, No.6, pp.720–727 (2006).
[24] Borromeo, R.M. and Toyama, M.: Optimization of SuperSQL Execution by Query Decomposition, *Proc. 5th International Conference on Advances in Databases, Knowledge, and Data Applications*, pp.65–70 (2013).
[25] Wolf, A., Borromeo, R.M. and Toyama, M.: A Data Retrieval Model Based on Independence Rules for SuperSQL, *Proc. 19th International Database Engineering & Applications Symposium*, pp.208–209 (2015).
[26] 金　哲，遠山元道：PPX：XML 整形出力のための出版言語，情報処理学会論文誌：データベース，Vol.3, No.4, pp.13–33 (2010).
[27] 慎　祥揆, 前田葉子, 遠山元道：ACTIVIEW：SuperSQL を用いた適応型 Web ビューの実現，情報処理学会論文誌：データベース，Vol.2, No.3, pp.112–129 (2009).
[28] Kiya, M., Goto, K. and Toyama, M.: GENERATE eHTML: Embedding SuperSQL Queries in HTML, *Proc. 19th International Database Engineering & Applications Symposium*, pp.224–225 (2015).
[29] Seto, T., Nagafuji, T. and Toyama, M.: Generating html sources with tfe enhanced sql, *Proc. ACM symposium on Applied computing*, pp.96–100 (1997).
[30] Ajax/API/jQuery，入手先 ⟨http://semooh.jp/jquery/api/ajax/jQuery.ajax/options/⟩（参照 2017-07-01）．
[31] Cache_Lite，入手先 ⟨http://pear.php.net/package/Cache_Lite/⟩（参照 2017-07-01）．
[32] CINEMA WORLD FREE CSS TEMPLATE, 入手先 ⟨http://www.free-css.com/free-css-templates/page138/cinema-world⟩（参照 2017-07-01）．

五嶋 研人　（学生会員）

平成 26 年慶應義塾大学大学院理工学研究科修士課程修了．現在，同大学大学院理工学研究科博士課程在学中．平成 26～29 年国立研究開発法人宇宙航空研究開発機構特別共同利用研究員．主にデータベースの研究に従事．人工知能学会，日本データベース学会，データベースシステム研究会，ソフトウェア工学研究会各学生会員．

木谷 将人

平成 28 年慶應義塾大学大学院理工学研究科修士課程修了．同年 4 月より東海旅客鉄道株式会社にて信号通信関連の職務に従事．

遠山 元道　（正会員）

慶應義塾大学理工学部情報工学科准教授．昭和 54 年慶應義塾大学工学部管理工学科卒業．昭和 59 年同大学大学院博士課程修了後，管理工学科助手，専任講師を経て現職．博士（工学）．平成 8 年 Oregon Graduate Institute 客員研究員．平成 10～13 年科学技術振興事業団さきがけ研究 21「情報と知」領域研究員．主にデータベースの研究に従事．電子情報通信学会，日本ソフトウェア科学会，IEEE，ACM 各会員．

（担当編集委員　川島 英之）