博士論文　　2019年度

Webアプリケーション開発のための埋め込み型SuperSQL

五嶋　研人
(学籍番号：81446729)

指導教員 教授 遠山元道

2020年3月

慶應義塾大学大学院理工学研究科
開放環境科学専攻

論文要旨

Web アプリケーションは，Web サーバー上で動作し，ブラウザを用いて利用されるアプリケーションである．Web アプリケーションは，DB プレゼンテーション部と入力 (form) 部からなる UI と，ビジネスロジックにより構成される．多数の Web アプリケーションが世に出ているなかで，公開されている Web アプリケーションの約 8 割が手続き型言語 PHP を利用して作成されている．

また，近年, スマートフォンやタブレットなどの Web を閲覧することのできる様々なデバイスが普及している．そこで, 数年前から注目され急速的に広がっているのが, レスポンシブデザイン（Responsive Web Design）と呼ばれるデザイン手法である．近年の Web アプリケーション開発においては，本デザインの適用が求められている．

しかし, 手続き型言語を利用した Web アプリケーション開発では，複雑なビジネスロジックも記述できるという利点がある反面，実装には複数のプログラミング言語を用いる必要があり，レイアウト等の変更時のコード量も多くなってしまうという問題点が存在する．また，レスポンシブデザインの実装には, 一般には HTML と CSS を用いたデザインの深い知識が必要であり，開発者の負担を大きくする要因となっている．

そこで本研究では，宣言型の DB プレゼンテーション言語である SuperSQL の実行を 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出来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 Web コードに対して SuperSQL クエリの埋め込みを可能とする埋め込み型 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て，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，及びキャッシュを利用した実行の分岐機構を実装した．また, Web のフロントエンド開発を支援するフレームワークである Bootstrap の方法論を活用し, SuperSQL クエリからレスポンシブな Web ページを生成する機構や, 結合子や反復子といった演算子を用いて構造の意図を記述することのできる SuperSQL の特長を利用することで, 通常の Web アプリケーション開発では困難な，画面幅に応じたコンテンツのサイズと位置の最適化を自動的に行う機構を提案, 実装した．

これらにより，Web アプリケーションにおける DB プレゼンテーション部，及び入力 (form) 部を SuperSQL クエリにより宣言的に実装する手法を実現した．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ーに対するXML形式での二次利用向けデータの提供を可能とし，キャッシュによる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．提案手法を用いたレスポンシブなWebアプリケーション作成では，HTML + JavaScript + PHPを用いた場合と比較してコード量は約3分の1となった. 自動生成されるレスポンシブレイアウトについては, 一致度が74.8%, 満足度は85.8%となり, 提案手法を用いることでレスポンシブなWebアプリケーションを開発する上での負担を減らすことが出来ることを示した.

Thesis abstract

Embedded SuperSQL for Web Application Development

A web application is an application that runs on a web server and is used via a Web browser. A Web application consists of a user interface(UI) consisting Database presentation part and Input(form) part, and business logic. Among many Web applications, about 80% of published Web applications are created using the procedural language PHP.

In recent years, various devices such as smartphones and tablets that can browse the Web have become widespread. A design technique called Responsive Web Design has been attracting attention and spreading rapidly for several years. In recent Web application development, implementation of this design is required.

However, Web application development using a procedural language has the advantage of being able to describe complex business logic, but it requires the use of multiple programming languages for implementation. And it also requires a large amount of code when changing layouts. In addition, implementing the responsive design generally requires deep knowledge of HTML and CSS design, which is a factor that increases the burden on developers.

In this paper, I propose the embedded SuperSQL as a set of PHP functions. The proposed embedded SuperSQL includes a declarative Web presentation language which is being researched and developed in Toyama Laboratory at Keio University. This method enabled to embed SuperSQL queries into an existing code including rich design HTML templates that is easily available to developers on the internet in recent years. Our approach applies the methodology of Bootstrap, a grid-based framework for front-end development, to generate responsive Web pages from SuperSQL queries. Furthermore, by using SuperSQL ' s features that enable the developer to describe the structure of the output document directly, I have developed a mechanism that can automatically adapt the size and the position of each of the contents on the website to a variety of device sizes. Using SuperSQL ' s features that allow the user to describe the structure of the output document directly, I have developed a mechanism that can automatically adapt the size and the position of each of the contents to a variety of device sizes.

By using embedded SuperSQL, the developer can declaratively implement the Database

presentation part and Input(form) part in Web applications using SuperSQL queries. Also, I propose the method to generate Web contents using XML, JavaScript, and CSS. The method developed in this research improved the maintainability of the generated Web contents and offered the XML data for secondary use to the Web user who visits the page. Moreover, the branch mechanism for execution using a cache was able to reduce the server load and accelerate the SuperSQL processing. In building a responsive dynamic website using the proposed method, the amount of code was reduced to about one-third compared to using HTML, JavaScript, and PHP. In an evaluation done in this research to study the behavior of automatically created layouts and manually created layouts, it was found that 74.8% of the automatically generated layouts behaved completely the same as the manually created layouts, and 85.8% of the layouts satisfied the developer ' s needs.

謝 辞

本研究を進めるにあたり, 御多忙な中, 長年に渡り最初から最後まで非常に御丁寧に御指導して下さっただけではなく本当に様々な面でサポートして下さった遠山元道教授に深く深く感謝いたします.

大変ご多忙な中，副査を務めて下さった高田眞吾教授，川島英之准教授，松谷宏紀准教授に深く感謝いたします.

また，ミーティングを通じて本研究に様々な意見をしてくれたSuper-SQL班, 研究室の皆様に深く感謝いたします.

本研究の基礎となる研究を行っていた先輩方に感謝いたします.

研究につまづいてしまったとき，様々な面で支えてくれた友人たちに感謝いたします.

最後に，これまで何不自由なく学生生活を送れるよう支援してくれた両親 五嶋晴記，五嶋惠理子，いつも応援して相談に乗ってくれたくれた妹たち 宮國桜子，五嶋麻里奈，そして様々な面で支えてくれ，公聴会の早朝にも起きて発表練習に付き合ってくれた妻 五嶋綺に心より感謝いたします.

2020年 2月 11日

目次

表 目 次

図 目 次

第 1 章

はじめに

1.1 背景

Web アプリケーションは，Web サーバー上で動作し，クライアント側から Web ブラウザを用いて HTTP アクセスにより利用されるアプリケーションである．Web アプリケーションは，データベースプレゼンテーション部と入力 (form) 部からなる UI と，ビジネスロジックにより構成される．ビジネスロジックの例としては，検索エンジンにおける検索アルゴリズムや地図アプリにおける経路検索アルゴリズムが挙げられる．実際に作成された Web アプリケーションの機能別コード割合を見てみると，5 つの小規模アプリ [2, 3, 4, 5, 6] の例では，DB プレゼンテーション部と入力 (form) 部が全体の約 28〜61%となっており，1 つの大規模アプリ [1] を例に取ると，全体の 15.6%がそれらに該当する．

Web アプリケーションの開発には，複数の手続き型言語が使用される．クライアントサイドの開発には HTML, JavaScript, CSS，サーバーサイドの開発には PHP が広く用いられており，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 79 %が手続き型言語 PHP を利用して作成されている [7]. 手続き型言語を利用した Web アプリケーション開発では，複雑なビジネスロジックも記述できるという利点がある反面，実装には複数のプログラミング言語を用いる必要があり，レイアウト等の変更時のコード量も多くなってしまうという問題点が存在する．

データベースに格納されているデータを Web（HTML），XML，PDF などの媒体ファイルとして出力することをデータベース出版/データベースプレゼンテーション（Database publishing / Database presentation）と呼ぶ．関係データベース [8] の誕生以前，Web ページ開発者はデータベースからの表の取得，取得した表の整形と媒体ファイルとしての出力のデータベース出版の全行程で手続き型言語を用いていた．関係データベースに誕生により，データベース出版の工程のなかのデータベースからの表の取得部分は非手続き的に SQL により行うことが可能となった．

宣言型・非手続き型言語 SuperSQL[9, 10, 11] は SQL の拡張言語であり，データベース出版言語として研究が続けられている．SuperSQL を利用した Web サイトの開発の大きなメリットは, 第一に, SuperSQL ワンソースの少ないコード量で Web サイトの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ので, 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．第二に, 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．SuperSQL では, 直感的にレイアウト構造を記述できる構文規則を用いているため, Web のフロントエンド側の様々な知識を要することなく Web コンテンツの生成ができ, 大幅な見た目のレイアウト変更などにも簡単に対応できるようになっている．

しかしながら SuperSQL による開発の現状の課題点として, まず, 従来の Generate

HTML（以下，独立型 SuperSQL, Standalone SuperSQL）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. 最近の Web の主流としては, HTML5 が取り入れられたことに伴い, 特にスタイルに関しては CSS として分離する傾向にある. さらに, Web コンテンツの生成に関しても, XML と JavaScript 等を用いて, クライアントサイドで生成するといったサイトも見られるようになってきている. 2 点目の課題点としては, 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ってしまうことである. 3 点目の課題点としては, SuperSQL の Web ページ生成機構においてこれまで開発されてきた主要な機能は, PC を想定とした Web ページを生成する機能とモバイル端末に最適化されたモバイル Web アプリケーションを生成する機能 [12] の二つであり, 機能によって生成可能なページの対象端末が分断されている状況にあることである.

また, 近年, スマートフォンやタブレット, その中間のファブレットなど, Web を閲覧することのできる様々なデバイスが登場し普及している. このインターネットを利用するための端末の多様化に対し, 以前はモバイル端末用と PC 用, といったように想定されるデバイス毎に HTML や CSS を用意し, 開発を行っていく方法が主流であった. しかし, Web ページの開発は, このような対象の端末によって Web サイトを別々に制作する方法では対応しきれなくなっている. そこで, 数年前から注目され, 急速的に広がっているのが, Ethan の著書「A List Apart」内で定義されたレスポンシブデザイン（Responsive Web Design）と呼ばれるデザイン手法である [13]. これは, 1 つの HTML ファイルと CSS ファイルを用意し, CSS 側に閲覧環境に合わせた分岐を用意することで各デバイスの画面サイズに応じて（つまり「レスポンシブ」に）レンダリングを切替え, 幅広い種類の端末表示に対応できるようにする Web デザインの手法である. 近年，Google 検索の約 61%は PC 以外からのアクセス [14] であり，世界トラフィック上位 100 サイトの約 54%[15]，有料 Web テンプレート販売大手 Theme Forest のテンプレートの約 51%[16] がレスポンシブデザインを採用している．このように，デバイスの多様化に伴いレスポンシブデザインの採用も増加傾向にあり，開発においてもレスポンシブデザインの適用が求められる．レスポンシブデザインの実装には，フロントエンド Web 制作のための手続き型の Web アプリケーションフレームワークである Bootstrap[17] が広く用いられている．レスポンシブデザインの実装は, ワンソースで幅広い閲覧環境に対応できるようになる利点がある反面, HTML と CSS を用いた Web デザインの深い知識が必要となり, 実装が複雑になるという問題点も存在する.

1.2 目的

本研究（以下，埋め込み型 SuperSQL, Embedded SuperSQL）では, 多くのユーザが利用する手続き型言語 PHP の関数として宣言型言語 SuperSQL の実行処理を組み込む機構を提案する．これにより，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 テンプレートや複雑なビジネスロジックを含んだ Web コード上での SuperSQL によるデータベースプレゼンテーション部，及び入力 (form) 部の実装を実現する．また，実行の分岐機構と生成物の分離を用いた Web コンテンツの生成を行う機構による，埋め込み型 SuperSQL 使用時のサーバ負荷の軽減・SuperSQL 処理（Web ページ表示）の高速化も可能とする．

さらに, SuperSQL を用いたレスポンシブな Web ページ生成機構を提案する. 本機構により, 今までの SuperSQL には不可能であった多様なレイアウトに宣言的な 1 つのクエリ (1 ソース) で対応できる Web アプリケーションの生成を実現し, 更に, 構造の意図を直接記述できる SuperSQL の構文規則の特徴を活かし, 各デバイスサイズに合わせたレイアウト生成を可能とする.

1.3 前提

開発コストには，開発の技術的難易度，開発する人の知識や経験の程度といった重要な考慮事項も挙げられるが，本研究では PHP や SuperSQL による Web 開発の基礎知識は既にあるものとし，開発ステップ数（コード量）のみを開発コストに関する評価対象とする．

本提案手法の利用者は，Web アプリケーションにおける DB プレゼンテーション部・入力 (form) 部の開発者を対象としている．

1.4 提案手法の適用条件・適用例

本研究は，下記を満たす Web アプリケーションのコンテンツ表示部・form 部の作成を対象としている．

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のデータは処理は，ストアドプロシージャ, ユーザー定義関数などのバックエンド SQL 処理系に依存

また，本提案手法は既存の Web アプリケーションコードとの協調を考慮して実装されており，例えば既存コード内の SQL との併用（例: 既存コード内の一部を提案手法で実装し，残りを SQL で実装）も可能である．

本手法を適用可能な箇所の例としては下記が挙げられる．

- Amazon[18] の商品一覧表示部・商品詳細表示部
- Google[19] の検索結果表示部
- Yahoo!ニュース [20] 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

1.5 貢献

埋め込み型 SuperSQL では，従来は手続き的に書かれていた DB プレゼンテーション部・入力 (form) 部の処理を宣言的に記述可能とし，宣言的クエリにより．生成されるコンテンツを Web ページ内の任意位置に表示させる仕組みによる開発コスト (コード量) の削減を実現する．レスポンシブデザイン生成では，デザイン変更が容易な 1 つの宣言的な SuperSQL クエリによる各デバイス画面幅に合わせた Web ページ生成の実現が貢献として挙げられる．

1.6 論文構成

本論文の構成は以下の通りである．第 2 章では既存手法である独立型 SuperSQL について述べ，第 3 章では提案手法である埋め込み型 SuperSQL の言語, 第 4 章では埋め込み型 SuperSQL の処理系, 第 5 章では埋め込み型 SuperSQL の用例について述べる．そして, 第 6 章では実験・評価について, 第 7 章では関連技術・関連研究について示す．最後に第 8 章で結論を述べる．

第 2 章

独立型SuperSQL

2.1 概要

独立型 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宣言型・非手続き型のデータベース出版/データベースプレゼンテーション言語である [9, 10, 11]．SuperSQL は，SuperSQL 利用者が既に持っている DBMS のデータを用いて，既存技術と比較してより少ない宣言的なコードでコンテンツ生成を行うことを目標としている．

2.2 クエリ

SuperSQL のクエリ（質問文）は SQL の SELECT 句を GENERATE <*medium*> <TFE> の構文をもつ GENERATE 句で置き換えたものである．ここで<*medium*>は出力媒体を示し, HTML, XML, PDF などの指定ができる．また<TFE>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, 反復子などのレイアウト指定演算子を持つ式である [21]．r_i を関係, P は SQL における条件文とすると, SuperSQL クエリは以下のように表せる．

GENERATE <*medium*>
<TFE>
FROM $r_1, r_2, ..., r_n$
WHERE P

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 (2.4 節) として指定して実行する．

2.2.1 TFE

TFE(Target Form Expression) は, SQL では SELECT 句に記述する属性名を, レイアウト演算子である結合子, 反復子と組み合わせることで出力結果の構造を指定する式である．さらに装飾子によって表のセルの幅や背景色などの詳細なカスタマイズが可能である．

2.2.2 結合子

結合子 (Connector) はデータベースから得られたデータをどの方向（次元）に結合するかを指定する二項演算子である．横方向を 1 次元, 縦方向を 2 次元, 深度方向を 3 次

元とし, それぞれ Connector の “C” と合わせて略記する. 表 2.1 にその種類と意味, 略記法を示す. なお, 表中の A と B は属性に限らずどんな TFE であっても構わない. ここで深度結合とは, 例えば HTML 出力ではハイパーリンクによる結合を表す. また時間結合とは, 時間軸方向に結合し, 動画が作成できる.

表 2.1: 結合子の種類と意味

結合子	意味	クエリ表記	略記法
,	横結合	A , B	C1
!	縦結合	A ! B	C2
%	深度結合	A % B	C3
#	時間結合	A # B	C4

2.2.3 反復子

反復子 (Grouper) は指定する方向に, データベースの値があるだけ繰り返して表示する単項演算子である. また反復子に関しても, Grouper の “G” と各次元方向を合わせて同様に略記する. 表 2.2 に反復子の種類と意味, そして略記法を示す. 結合子と同様, 表中の A はどんな TFE であっても構わない. 例えば,

[学籍番号 , 評点]!

と記述することで, 横方向に連結された学籍番号とその生徒の評点をインスタンス数だけ縦方向に反復して出力する.

また反復子は単に構造を指定するだけでなく, 反復したデータとそれをグルーピングするデータのネストの関係によって属性間の関連を指定することができる. 例えば

[科目名]! , [学籍番号]! , [評点]!

表 2.2: 反復子の種類と意味

反復子	意味	クエリ表記	略記法
[ ],	横反復	[A],	G1
[ ]!	縦反復	[A]!	G2
[ ]%	深度反復	[A]%	G3
[ ]#	時間反復	[A]#	G4

表 2.3: 複合反復子の構文と動作

構文	動作
[*A*],*number*!	横に *number* 個結合し, 次の行に改行
[*A*]!*number*,	縦に *number* 個結合し, 次の列に改行
[*A*],*number*%	横に *number* 個結合し, ページ切り替え
[*A*]!*number*%	縦に *number* 個結合し, ページ切り替え
[*A*],*number1*!*number2*%	横に *number1* 個結合し, 次の行に改行し, *number2* 行分出力後、ページ切り替え
[*A*]!*number1*,*number2*%	縦に *number1* 個結合し、次の列に改行し, *number2* 列分出力後、ページ切り替え

とすると, 単に各々の一覧が表示されるだけで互いの関連は失われるが,

[科目名 ! [学籍番号 , 評点]!]!

と反復子を入れ子状にすることで, その科目における学生の評点一覧が表示される.

2.2.4 複合反復子

複合反復子は, 反復子を組み合わせ, 指定する個数分値を指定する方向に繰り返し表示し, 指定する方向に改行を行うものである. さらに, 深度（%）演算子を用いることで, ページ切り替えも行うことが可能である. 表 2.3 に複合反復子の構文と動作を示す. 表中の A はどんな TFE であっても構わない.

2.2.5 装飾子

SuperSQL で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などの情報を付加できる. 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 (項目名 = 値) として指定する. 複数指定するときは各々を “,” で区切る. 例えば, 科目名の背景を赤にして, セル内の中央に文字列を配置したい場合は以下のように記述する.

[科目名@{bgcolor='red', align='center'} ! [学籍番号 , 評点]!]!

装飾子はこのように属性に指定するだけでなく, 反復子に対して指定することも可能である.

2.2.6 関数

SuperSQL における関数は, 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. 関数の記述は

関数名 (引数 1[, 引数 2[, ...[, 引数 N]...]])

の形式で行う. 既存のシステムでは画像を出力したい場合に用いる image 関数, HTML 出力 におけるハイパーリンクを動的に構築する invoke 関数, HTML 出力におけるハイパーリンクを静的に構築する link 関数やクエリを細分化する embed 関数などが利用可能である.

2.2.7 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 TFE1 の内容が表示され，偽のときは TFE2 が表示される.

if(条件式) then ($TFE1$) else ($TFE2$)
(条件式)? $TFE1 : TFE2$

2.3 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. その記述は以下のとおりである.

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 (2.4 節) として指定されたファイル を基にして行う.

2.4 実行引数

SuperSQL の実行の際に実行オプションとして引数を指定することで, 様々な条件での実行を行うことが可能である. 以下に実行の引数の一例を示す.

- -f [ファイルのパス（.ssql ファイル）]
  - 指定したクエリファイルに対して SuperSQL を実行
- -c [設定ファイル (2.3) のパス（config.ssql 等）]
  - SuperSQL の設定を指定したファイルから読み込み
- -driver [データベースのドライバ名]
  - 指定したデータベースドライバを使用して実行
- -db [データベース名]
  - 指定したデータベース名を使用して接続
- -host [ホスト名]
  - 指定したホストを使用して実行
- -u [ユーザ名]
  - 指定したユーザ名を使用して接続
- -p [パスワード]
  - 指定したパスワードを使用して接続
- -d [出力先ディレクトリ]
  - 指定した出力先に結果を生成
- -o [生成するファイル名]
  - 指定したファイル名で出力結果を生成

通常の SuperSQL の実行には-f オプションを用いて, 実行を行う SuperSQL クエリを指定している.

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 から引数 p を指定している場合，設定ファイル (2.3 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 (2.2 節) が置かれているディレクトリとなる.

2.5 特徴

ここでは, 提案手法の説明に入るにあたり, SuperSQL の特徴を説明する. なお, 本節内の図と例で使用している映画の画像と解説は, Yahoo!映画から引用している[1]. SuperSQL は, 非常に少ないコード量で Web コンテンツを生成することが可能である. 例えば, 図 2.1 のようなシンプルな表を実現したいと考えたとき, PHP, SQL, HTML を利用すると, 約 20 行ほどとなり, Web サイト開発者は PHP 以外にも, SQL, HTML 等の知識が必要となる. 一方で, SuperSQL を用いると,

```
クエリ 1
GENERATE HTML {
[m.title, image(image, path="img"), g.name, c.name, m.abst]!
} FROM movie m, genre g, company c
WHERE m.genre = g.id AND m.company = c.id;
```

の 4 行のクエリで実現することが可能であり, Web サイト開発者は, HTML 等の知識がほとんどなくても利用することが可能である.

トイ・ストーリー		アニメ	ピクサー・アニメーション・スタジオ	カウボーイ人形のウッディはアンディ少年の大のお気に入り。だがそれも誕生日プレゼントでアクション人形バズ・ライトイヤーを手にするまでの事だった。NO. 1の座を奪われたウッディは何とかバズをこらしめようとするが、バズはバズで自分が本物のスペース・レンジャーだと思い込んでいる有り様。そんな二人がふとしたいざこざから外の世界に飛び出してしまう。なんとか我が家へ帰還しようとする二人だが、なんとアンディの隣に住む悪ガキのシドに捕まってしまった……。
ファインディング・ニモ		アニメ	ピクサー・アニメーション・スタジオ	舞台はオーストラリアの美しい海。カクレクマノミのマーリンは、妻との間に生まれた卵の世話をしながら生活しており、目の前に迫った子どもたちの孵化を今や遅しと待ち構えていた。とても幸せな日々を送っていたマーリンだったが、毒を持つオニカマスに襲われてしまい、卵と妻は姿を消してしまった。たった一つだけ残った卵から生まれた子どもにニモと名付け、愛情たっぷりに育てた。ニモは過保護なマーリンにうんざりしながらも学校に通っていたが、漁をしていた人間に捕らわれてしまった。マーリンはニモを奪還すべく、ナンヨウハギのドリーと共に旅に出た……
天使にラブソングを		コメディ	タッチストーン・ピクチャーズ	ウーピー・ゴールドバーグの人気を不動の物にしたミュージック・コメディ。とある殺人現場を目撃したために、組織に命を狙われるようになった売れないクラブ歌手が、裁判の日まで修道院でかくまわれるハメに。しかし、元々下町で下品に育った彼女がそんなに神聖にできるはずもなく、やがて、聖歌隊をゴスペル風に改造し……

図 2.1: クエリ 1 により生成される表

さらに, 図 2.1 を図 2.2 のようにレイアウトの変更を行いたい場合, PHP, SQL, HTML を利用すると約 40 行となり, レイアウト変更前と比較して約 2 倍となる上, コード全体を修正する必要があるが, SuperSQL では,

[1] 出典：https://movies.yahoo.co.jp/

クエリ 2
GENERATE HTML {
[{image(image, path=”img”), {m.title! g.name! c.name}}! m.abst],3!
} FROM movie m, genre g, company c
WHERE m.genre = g.id AND m.company = c.id;

の 4 行で実現することができる. 上記のクエリを見てわかるように, コードの変更量も非常に小さく, コードの変更箇所も少ない.

図 2.2: クエリ 2 により生成される表

そして, 図 2.1 を図 2.3 のように入れ子の表を組みたい場合, PHP, SQL, HTML を利用すると約 50 行となってしまう上, for 等を使用したネストのコードを記述する必要がある. しかし, SuperSQL では,

クエリ 3
GENERATE HTML {
[c.name, [g.name, [image(image, path=”img”), m.title, m.abst]!]!]!

} FROM movie m, genre g, company c
WHERE m.genre = g.id AND m.company = c.id;

で実現することができる. 先ほどのレイアウト変更同様, コードの変更量も非常に小さく, コードの変更箇所も少ないことがわかる.

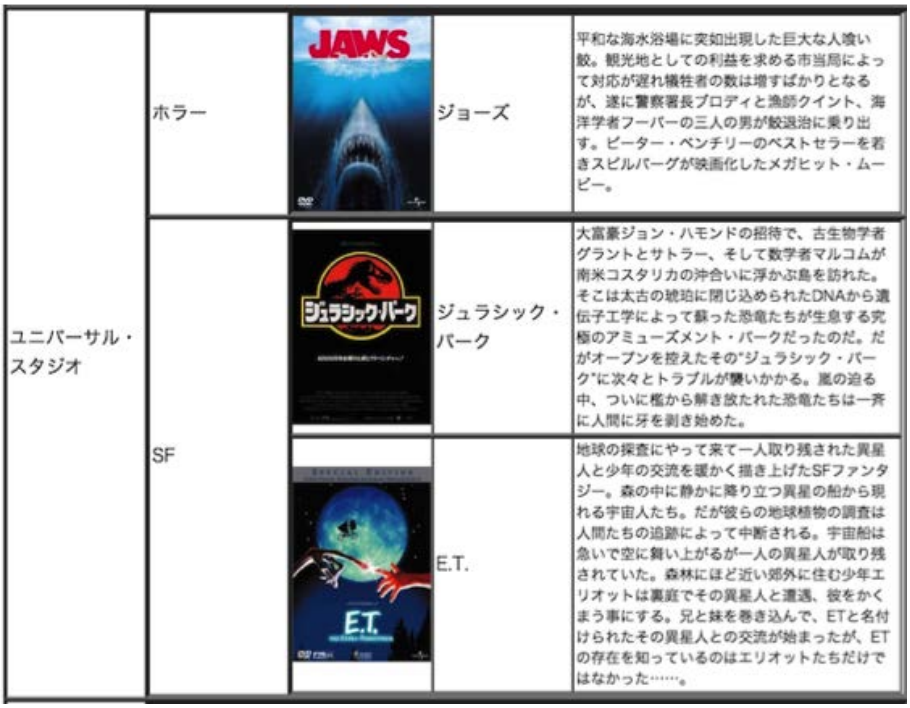


図 2.3: クエリ 3 により生成される表

このように, SuperSQL を利用することで簡単に Web コンテンツの生成を行うことが出来るだけではなく, そのレイアウトを変更したいといった場合でも, 非常に少ないコード量の変更で済むのが大きな特徴の一つである.

2.6 独立型 SuperSQL を用いた Web アプリケーション生成

2.6.1 デスクトップ向けの Web アプリケーション生成

SuperSQL には, GENERATE HTML という, HTML の生成機構がある. この機構は, 主に HTML の table タグを用いた静的な閲覧専用の HTML を生成することに特化している. 有薗 [22] により提案された Web アプリケーション開発のための SuperSQL の拡張では, 限定的なレイアウト・機能によるデスクトップ向けの Web アプリケーションの生成が可能となった.

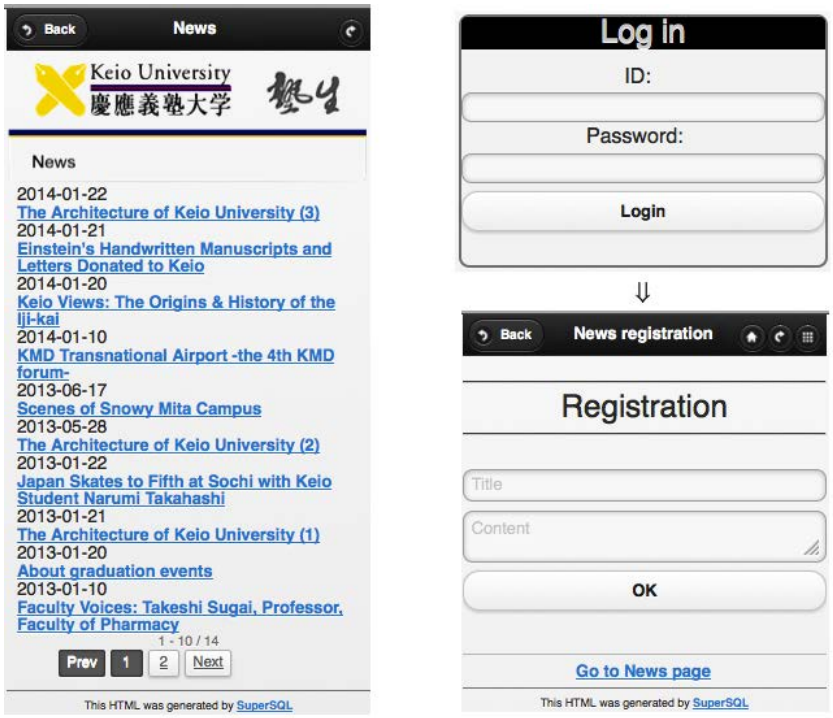


図 2.4: SuperSQL を用いて生成したモバイル Web アプリケーションの一例

2.6.2 モバイル端末向けの Web アプリケーション生成

先行研究 [12] では，Web の閲覧端末としてのモバイル端末の普及に対応する形として，モバイル端末を対象とした Web アプリケーション生成機構 (GENERATE Mobile_HTML5) を提案，実装した．既存技術と比較してより短いコード量でモバイル向けの Web アプリケーションを生成するために，以下の変更を行った．

- サーバサイドで動くモジュールを同時に生成する機構を実装
- クエリ内で指定された特定箇所のみの動的表示機構を実装
- レイアウトに使用されるタグとして，従来の table タグに加えて div タグを追加
- タッチ操作に最適化したウェブを開発するためのフレームワークである jQuery Mobile を用いて全体の基盤となるデザインや挙動を統一化

これらの変更点により，モバイル端末向けの表示限定，かつ限定的なレイアウトではあるが，フォームなどを用いたデータの入力機能やログイン機能など，データベースとのやり取りを基盤とする機能を持たせたモバイル Web アプリケーションの生成を実現した．図 2.4 にこの機能を用いたモバイル Web アプリケーションの一例を示す．

上記のフォーム等の生成機構は，関数形式 (2.2.6 項) を用いたフォーム生成となっておりそのレイアウトやデザインは固定である．本稿の提案手法では上記をベースとした拡張・改良を行い，可変レイアウト・デザインによるフォーム生成を実現している．

第 3 章

埋め込み型SuperSQL:言語

3.1 概要

本提案手法である埋め込み型 SuperSQL は, 既存技術に比べより短い宣言的なコードでの Web コンテンツの生成を実現する．既存技術では大幅なコード量の変更を伴う 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，宣言型言語である独立型 SuperSQL 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 を埋め込む機構を提案する．また, 既存コードへの埋め込みに伴い, 近年の Web の分離傾向を取り入れ, 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 は既にあるデータベース（例: 映画会社の映画情報の DB, 旅行サイトの旅行プランの DB など）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いる．クエリ（2.2 節）と設定ファイル（2.3 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.

3.2 独立型 SuperSQL と提案手法の違い

本節では，独立型 SuperSQL と提案手法の埋め込み型 SuperSQL の違いについて述べる．図 3.1 にその違いを示す．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（2.2 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，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 3.2 に示す.

3.3 埋め込み型 SuperSQL

提案手法は, Web ページや Web アプリケーション全体を生成するのではなく, 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, 閲覧

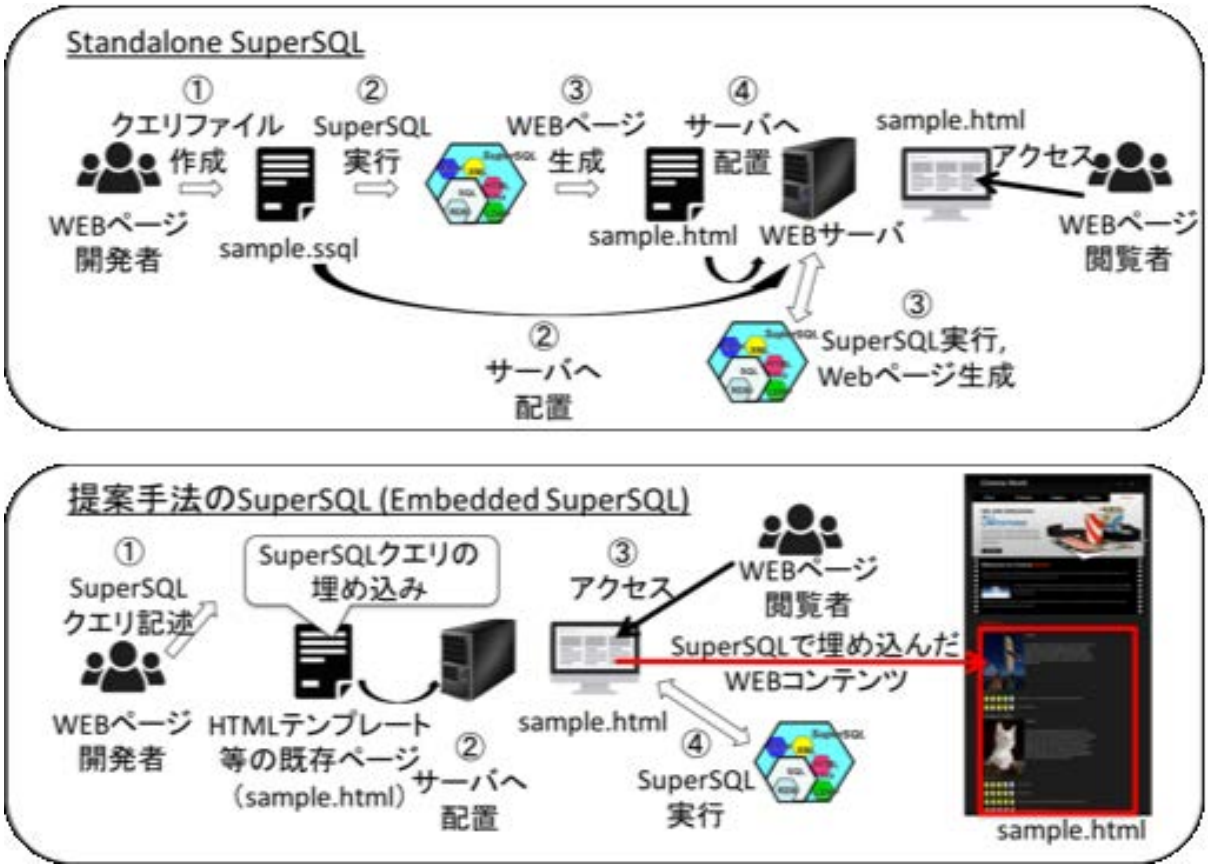


図 3.1: 独立型 SuperSQL と提案手法の埋め込み型 SuperSQL の違い

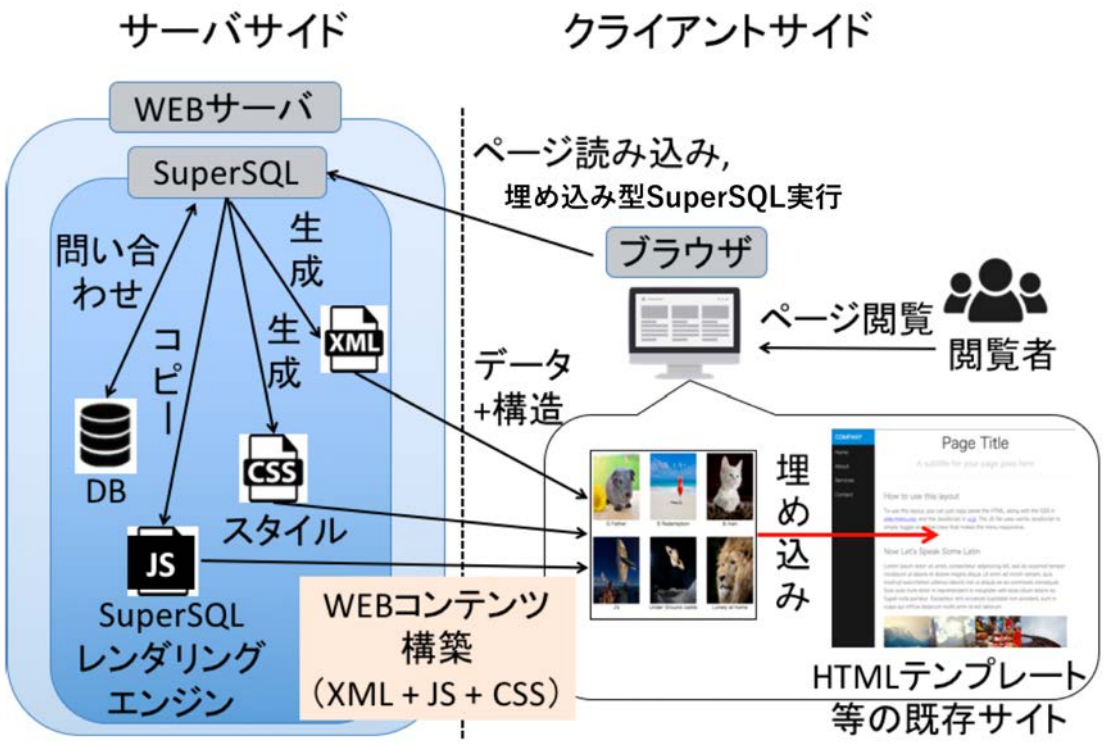


図 3.2: 提案手法のアーキテクチャ

者によるページアクセス時に SuperSQL を実行するといった動的スクリプト言語的な処理を行う必要がある．このことから，提案手法では動的スクリプト言語である PHP に SuperSQL 実行処理の組み込みを行った．Ruby, Python, JavaScript, Perl など様々な動的スクリプト言語があるなかで PHP を選択した理由としては, 動的 Web サイトの開発に特化していることと, 近年の動的 Web サイトの約 82%が PHP を利用して開発されており, 利用ユーザが非常に多いことなどが挙げられる．

また，本手法では，Web アプリケーションの機能として使用頻度の高い，Web アプリケーションの UI 部の入力を司る form の機能，及び近年の Web 閲覧者端末の多様化に伴い採用が増加傾向にあるレスポンシブデザインを生成する機能を実装した．

3.3.1 実行関数

PHP 上で SuperSQL を動作させるために実装した PHP の関数を表 3.1 に示す．提案手法を利用する場合は, 表 3.1 に示す 4 つの PHP の関数を使用する．

表 3.1: 提案手法で実装した PHP の関数

関数名	機能	引数	必須かどうか	使用例
ssql_setConfig()	SuperSQLの設定の読み込み	第一: 設定ファイルのパス	✓ (同一ページ内では1回)	ssql_setConfig('./config.ssql')
		第一: ドライバ名 第二: DB名 第三: ホスト名 第四: ユーザ名 第五: パスワード		ssql_setConfig('postgresql', 'testDB', 'localhost', 'ユーザ名')
ssql_exec()	SuperSQLの実行	第一: SuperSQLクエリ 第二: キャッシュ保持時間 (秒, デフォルト: 0秒,省略可) 第三: DB問い合わせ間隔(秒, デフォルト: 0秒, 省略可)	✓	ssql_exec('SuperSQLクエリ') ssql_exec('SuperSQLクエリ', 3600) ssql_exec('SuperSQLクエリ', 3600, 60)
ssql_cacheTime()	キャッシュを保持する時間の設定	第一: キャッシュ保持時間 (秒)		ssql_cacheTime(3600)
ssql_selectDB_interval()	DBへの問い合わせ間隔の設定	第一: DB問い合わせ間隔の設定 (秒)		ssql_selectDB_interval(60)

表 3.1 の最初の二つの関数は必須であり，次の二つはオプションである．ssql_setConfig 関数は SuperSQL の設定（2.3 節）の読み込み，ssql_exec 関数は SuperSQL クエリ（2.2 節）の実行に用いる．また，ssql_cacheTime 関数はキャッシュを保持する時間を，ssql_selectDB_interval 関数は DB への問い合わせ間隔をともに秒で指定する．これら下 2 つの指定は，ssql_exec 関数の第二, 第三引数に記述することも可能である．同関数への指定により，クエリ単位でのキャッシュ保持時間等の変更を可能としている．指定が無い場合のデフォルト値は，ともに 0 秒となる．

3.3.2 plink，glink 関数と parameter 句

埋め込み型 SuperSQL では，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 句も併せて実装した．plink 関数，glink 関数の仕様を表 3.2 に示す．

表 3.2: plink 関数と glink 関数の仕様

関数名	機能	引数	使用例
plink()	POST形式での値の受け渡しを行うリンクの作成	第一: リンクとなる属性 第二: 値の受け渡し先ファイル名 第三以降:受け渡す値のキーとなる属性	従業員名をリンクとして表示し，従業員IDによりリンク先を切り替える plink(e.name, 'pageB.html', e.id)
glink()	GET形式での値の受け渡しを行うリンクの作成	第一: リンクとなる属性 第二: 値の受け渡し先ファイル名 第三以降:受け渡す値のキーとなる属性	

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，関数の第三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．

PARAMETER 受け取る値の属性
GENERATE HTML
<TFE>
FROM テーブル
WHERE 条件式

3.3.3 入力 (form) 関数

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，Web アプリケーションの UI 部の入力を司る form の機能を実装した．

この機能は下記 GENERATE から始まるクエリの form タイプ部分に指定できる 3 つの装飾子「登録: insert, 更新: update, 削除: delete」と，制限付き TFE 部分へ指定できる表 3.3 に示す各装飾子により構成される．制限付き TFE 部分では，演算子として横結合 (,)，縦結合 (!)，装飾子としては既存のデザイン用装飾子も記述可能である．

GENERATE HTML
<制限付き TFE> @{ form タイプ }

FROM テーブル
WHERE 条件式

表 3.3: 制限付き TFE 内で使用する form 作成用装飾子

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}

指定された装飾子により，例えば，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 3.3 中の装飾子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を表中の装飾子 sql として実装している．

3.3.4 レスポンシブレイアウト生成

レスポンシブレイアウト生成は，従来の SuperSQL では不可能であった, レスポンシブデザインに 1 ソースで対応できる Web ページの生成を実現する新たな機構である. 提案機構は, 2.5 節で述べた既存技術に比べてより短いコードでの Web コンテンツの生成を実現する．

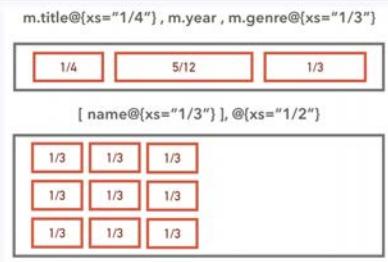
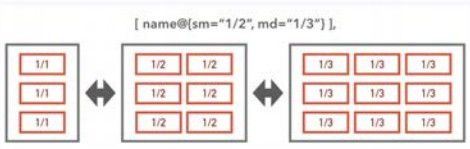
本機能は，下記に述べるクエリに装飾子を指定してレスポンシブデザインを生成する方法と，自動でレスポンシブデザインを生成する方法の 2 つにより構成される．本機能は，Web コンテンツのレイアウトの変更も容易に行うことができる宣言型 SuperSQL の特徴を利用しながら, 通常では複雑な HTML と CSS の記述が必要となるレスポンシブレイアウトを SuperSQL の構造演算子（結合子, 反復子）と装飾子のみを用いて記述可能にすることで, 複雑になりがちなレスポンシブ Web アプリケーションの制作の負担を軽減する手法を提案する.

指定生成：クエリに指定してレスポンシブデザインを生成

ここでは, レスポンシブなレイアウトを装飾子指定により生成するために導入した装飾子と, それらを用いて SuperSQL クエリ上でレイアウトを構成するためのクエリの記述方法について説明する.

本レスポンシブデザイン生成は，メディアとして新たに導入した「ResponsiveHTML」を使用する．また，従来の SuperSQL では, 要素の横幅や縦幅を指定する際には, width 装飾子を使用し, `@{width='200px'}`といったように, 絶対単位を用いて固定幅で指定していた. ページ要素に絶対単位を用いることは, どのような場合でもレイアウトを変えたくない場合は有効であるが, 本研究が目指しているレスポンシブデザインの実現には適していない. そこで提案機構では, 画面幅に合わせた要素のサイズ指定を実現するために, Bootstrap[17] の方法論をベースに表 3.4 に述べる xs, sm, md, lg, xl という 5 つの装飾子を新たに導入した. Bootstrap の詳しい説明については次章で述べる.

表 3.4: 装飾子指定によるレスポンシブデザイン生成に用いる装飾子

装飾子	値	機能	使用例
xs	行(row)における指定箇所の割合 / 行(row)における任意の等分	画面幅 < 544px のときの行(row)におけるコンテンツ表示幅の割合指定	
sm		画面幅 < 768px のときの行(row)におけるコンテンツ表示幅の割合指定	
md		画面幅 < 992px のときの行(row)におけるコンテンツ表示幅の割合指定	
lg		画面幅 < 1200px のときの行(row)におけるコンテンツ表示幅の割合の指定	-
xl		1200px ≦ 画面幅 のときの行(row)におけるコンテンツ表示幅の割合の指定	-

表中の装飾子は, 多様な画面幅における要素の幅を相対的な割合で指定するための装

図 3.3: ResponsiveHTML の生成物の例 1

飾子であり, それぞれの装飾子は, 次章の表 4.2 に示される挙動になるような CSS を生成する. サイズ指定は, 分数を用いて相対的に表現する.

これらの装飾子を利用したクエリの書き方を例を用いて説明する. 下記に示すクエリ例 1 は, 猫の情報が格納されたデータベースから, 猫の画像 (image), 名前 (name), 種類 (breed) の情報を取得し, 表示するクエリである. このクエリによって生成されるページを図 3.3 に示す. このクエリでは, 縦結合で結ばれた猫の名前 (name) と種類 (breed) が猫の画像 (image) と横結合で結ばれており, 全体が縦の反復子によって繰り返し表示されるようになっている. 猫の画像 (image) には`@{xs="2/5"}`が, 同じ階層にある猫の名前 (name) と種類 (breed) を中括弧で囲った部分には`@{xs="3/5"}`が指定されている. ここで使われている装飾指定は`@{xs}`であることから, 同じ階層にある猫の画像 (image) と情報（name と breed）は, それぞれ 2/5, 3/5 の割合を保ちながら, どのような画面幅であっても横並びで表示される.

ResponsiveHTML のクエリ例 1

```
GENERATE ResponsiveHTML
  [ image(c.image)@{xs="2/5"} , {c.name ! c.breed}@{xs="3/5"} ]!
FROM cat c
```

これに対し, 装飾指定に`@{sm}`を使用したクエリ例 2 のようなクエリを実行した場合は, 画面幅に応じて要素の配置とサイズが切り替わるレスポンシブなレイアウトとなる. 生成されるページを図 3.4 に示す. この例では表示の切り替わるブレイクポイントとして `sm` を指定しているため, 画面幅が 768px 以上のときは, 指定された割合を保ちながら横並びで表示され, 画面幅が 768px を下回った場合は, 割合の指定は放棄され要素が縦並びに配置される.

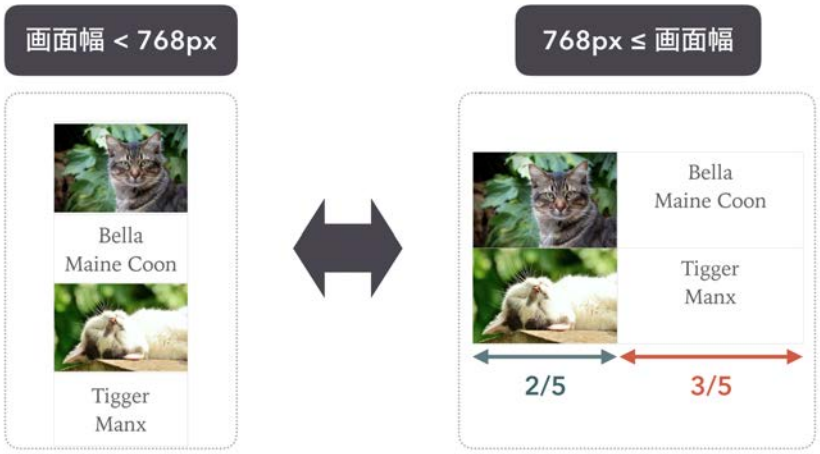


図 3.4: ResponsiveHTML の生成物の例 2

ResponsiveHTML のクエリ例 2

```
GENERATE ResponsiveHTML
  [ image(c.image)@{sm="2/5"} , {c.name ! c.breed}@{sm="3/5"} ]!
FROM cat c
```

このように, 提案機構では, 従来の SuperSQL に備わっていた結合子や反復子などの構造演算子を用いてレイアウトを構成する仕組みと, 次章で述べるブレイクポイントや可変グリッドなどの考え方を用いてレスポンシブデザインを実現している Bootstrap の方法論を組み合わせることで, 直感的にレスポンシブなレイアウトを記述することができるようになっている.

ここから先は, 更に細かい仕様について説明する. 例では, 分かりやすさのためクエリを省略し, クエリの TFE の部分だけを示しているほか, 生成されるコンテンツについても簡略化を行っている.

- クエリに横結合や横反復が含まれない場合 (図 3.5)
  - サイズ指定がされた部分をそのサイズで出力
  - サイズ指定がない部分は横幅 100%で出力
- クエリに横結合が含まれる場合
  - サイズ指定がない場合 (図 3.6)
    - 各要素の横幅は, 全体を要素数で等分した割合を割り振り
    - 画面幅に応じたレイアウトの変化はなし
  - サイズ指定が全ての要素にある場合 (図 3.7)
    - 指定どおりの割合で出力
  - サイズ指定が一部の要素にある場合 (図 3.8)

    - 指定されている値を全体 (通常は 1) から差し引いて, 残りの要素に割り振る
    - 指定されている値の合計が 1 を超える場合は, その合計より大きく, 最も近い整数から差し引く

- クエリに横反復が含まれる場合
  - サイズ指定がない場合 (図 3.9)
    - 反復されるデータの個数で全体を等分
  - 反復子の内側の要素にサイズ指定がある場合 (図 3.10)
    - 指定された割合で出力
    - 1 行に収まらない場合は次の行に繰り越して表示
  - 反復子の内側の要素に複数のサイズ指定がある場合 (図 3.11)
    - 指定された割合で出力
    - 指定されたブレイクポイントに合わせて要素の並び方が組み変わる
  - 反復子に (外側に) サイズ指定がある場合 (図 3.12)
    - 反復されるデータ全体が指定された割合の中で表示される

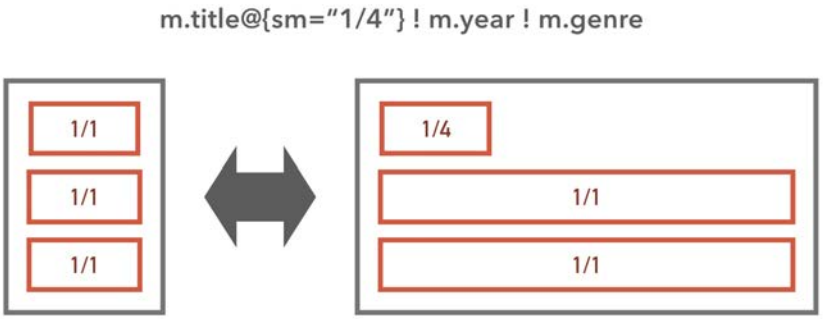


図 3.5: クエリに横結合や横反復が含まれない場合の例

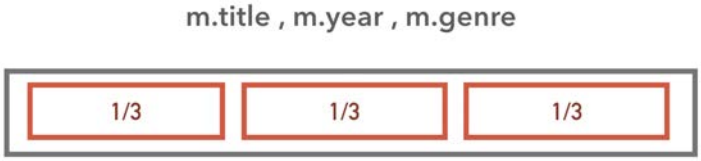


図 3.6: 横結合の例 (サイズ指定がない場合)

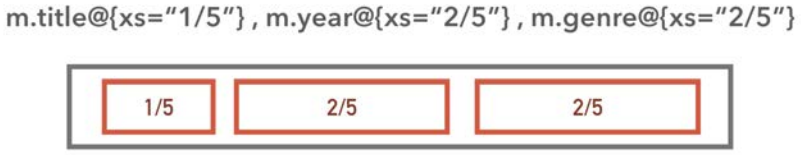


図 3.7: 横結合の例 (サイズ指定が全ての要素にある場合)

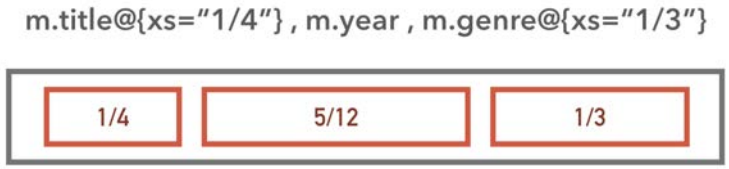


図 3.8: 横結合の例 (サイズ指定が一部の要素にある場合)

図 3.9: 横反復の例 (サイズ指定がない場合)

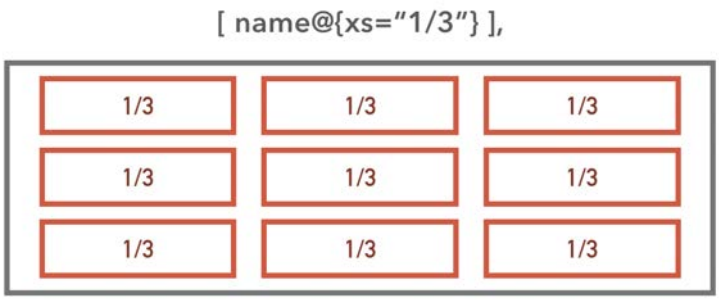


図 3.10: 横反復の例 (反復子の内側にサイズ指定がある場合)

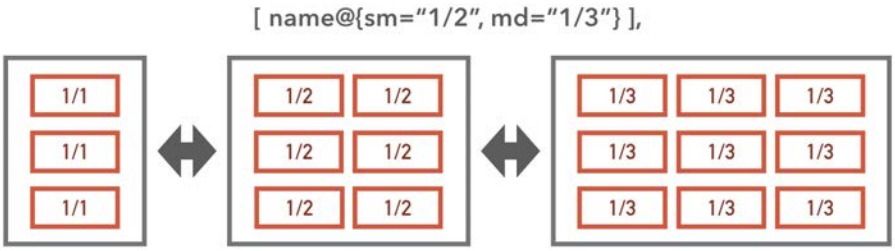


図 3.11: 横反復の例 (反復子の内側に複数のサイズ指定がある場合)

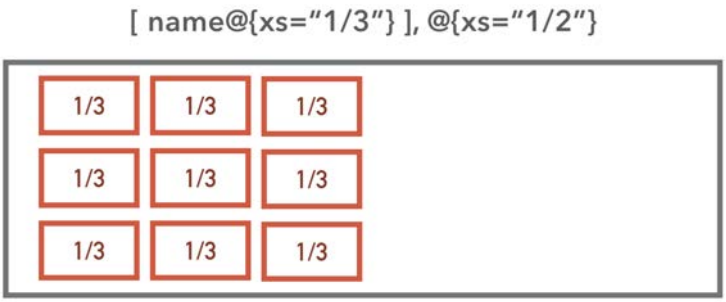


図 3.12: 横反復の例 (反復子にサイズ指定がある場合)

自動生成：レスポンシブデザインの自動生成

本研究では, これまでに説明してきたユーザ主導のレスポンシブレイアウトの生成機能に加え, 構造の意図を直接記述できる SuperSQL の構文規則の特徴を利用した, レスポンシブレイアウトの自動生成機能を提案, 実装している. この機能の利用には, 前提として, 基準レイアウトを生成する SuperSQL クエリが必要となる. 基準レイアウトは, 提案手法の仕様上, デスクトップ環境を想定としたレイアウトで, かつ, 画面幅によって配置の変わらないレスポンシブでないものであることが望ましい. そのため, ユーザには, 画面幅によって挙動が変わらない xs 装飾子のみを用いて, 始めに基準レイアウトを作成してもらう必要がある. 作成した基準レイアウトを元にレスポンシブなレイアウトを生成するには, GENERATE 句の TFE 全体を中括弧で囲み, `@{responsive}`といった形で responsive 装飾子を TFE 全体に指定する. これにより, クエリ実行時にレイアウト自動生成機構が動き, 各デバイスの画面サイズに最適化されたレスポンシブ Web アプリケーションが生成される.

よって, 提案機構が想定しているユーザのワークフローをまとめると以下のようになる.

1. xs 装飾子のみを用いて, デスクトップ PC 環境 (横幅 1200px) を想定したレイアウトを作成
2. GENERATE 句の TFE に`@{responsive}`装飾子を付与し, レスポンシブなレイアウトを作成

上記 2 で述べられている装飾子について表 3.5 に示す.

表 3.5: レスポンシブデザインの自動生成に用いる装飾子

装飾子	値	機能	制限	使用例
responsive	topdown bottomup	トップダウン/ボトムアップでレスポンシブデザインを自動生成	指定はTFE全体	GENERATE ResponsiveHTML [image(c.image), {c.name ! c.breed}]!@{responsive='topdown'} FROM cat c;

その他の機能：関数

GENERATE ResponsiveHTML でデザインの基盤として導入している Bootstrap では, レスポンシブを想定としたレイアウト構成を体系化しているだけでなく, 汎用的で統一された基本デザインのテンプレートの提供も行っている. 提案手法では, こうした統一感のあるデザインの UI を SuperSQL から簡単に作成できるように, 複合演算子を使用して生成可能なページネーション用の UI など, 従来の SuperSQL に備わっていた機能によって生成される Web コンテンツの出力内容を見直した. また, レスポンシブなナビ

図 3.13: UI コンポーネントの例

ゲーションバーや, ポップアップウィンドウなどを生成する関数を新たに実装した. 図 3.13 に提案機構で生成可能な代表的な UI コンポーネントの例をまとめたものを示す.

navbar 関数

navbar 関数を用いることにより, 図 3.3.4 のようにページにナビゲーションバー (メニューバー) を作成することが可能となる. navbar の書き方を下に示す. 関数の最初の引数にはナビゲーションバーの一番左に表示されるページのタイトルと, アプリケーションのホームとなるページへの URL を記述する. 2 つ目以降の引数には, ナビゲーションバーに表示するそれぞれのメニュー項目と, そのメニュー項目に対応するページへ遷移するための URL を記述する. 引数の数に限りはなく, 指定された数だけメニュー項目が生成される.

```
navbar(
  "Title: 'Home URL' " ,
  ["Menu 1: 'Menu 1 URL'", "Menu 2: 'Menu2 URL'",... ]
)
```

```
dropdown(
  "Dropdown Menu Name" ,
  ["Menu 1: 'Menu 1 URL'", "Menu 2: 'Menu2 URL'",... ]
)
```

また, navbar 関数の 2 つ目以降の引数には, 関数内で関数を呼び出す形として, ドロップダウンメニューを作成する dropdown 関数が使用可能である. dropdown 関数の書き方を上に示す. dropdown 関数では, 1 つ目の引数にはそのドロップダウンメニューを開くボタンに表示される文字列を書く. 2 つ目以降の引数の書き方は navbar と共通して

おり, ドロップダウンメニューに表示するそれぞれのメニュー項目と, そのメニュー項目に対応するページに遷移するための URL を記述する. navbar 関数と dropdown 関数を組み合わせてナビゲーションバーを生成するためのクエリの例を以下に, 生成されるナビゲーションバーを図 3.14 に示す. 生成されるナビゲーションバーは 768px(sm のブレイクポイント) より小さい画面幅に表示された際は, 折りたたまれ, 画面右上のハンバーガーボタンで開閉してメニュー項目にアクセスできるようになっている. 図 3.14 の上部は, 画面幅がブレイクポイント以上の場合のナビゲーションバーの表示を, 下部は画面幅がブレイクポイントを下回った場合のナビゲーションバーの表示を示している.

navbar 関数のクエリ例

```
GENERATE ResponsiveHTML
  navbar(
    "Cat Cafe: '../home'",
    "News: '../news'",
    "Menu: '../menu'",
    "Recruit: '../recruit'",
    "Access: '../access'",
    dropdown("About",
      "About Us: '../aboutus'",
      "Contact: '../contact'"
    )
  )
FROM ...
```

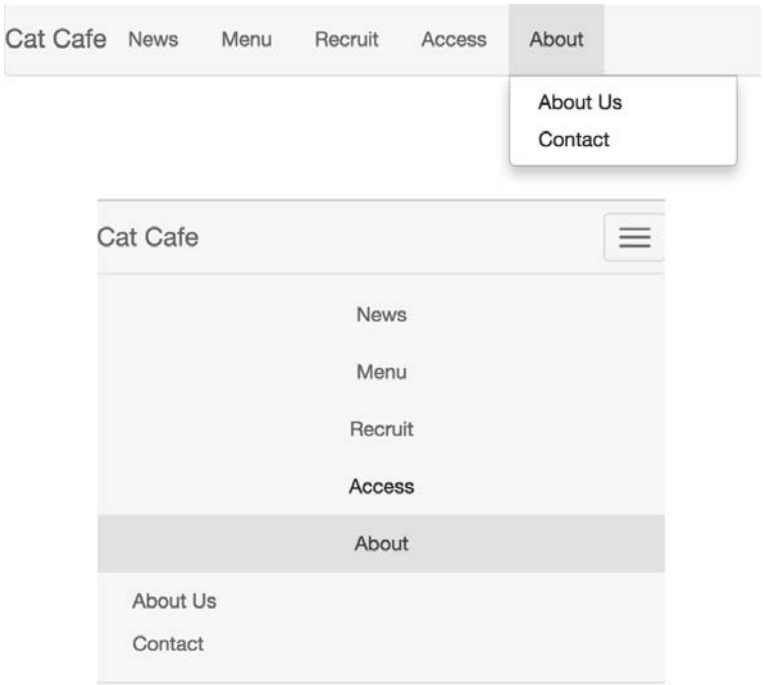


図 3.14: navbar 関数で生成されるナビゲーションバーの例

表 3.6: ポップアップ関数の関数名

関数名	説明
popup	テキストクリック式ポップアップ
popup_button	ボタンクリック式ポップアップ
popup_image	画像クリック式ポップアップ

ポップアップ関数

本関数を用いることで, Web ページ中にポップアップを埋め込むことが可能となる. 本関数の関数名を表 3.6 に示す.

```
ポップアップ関数の関数名 (
  ”ポップアップ元の Title/Image Path” ,
  ”ポップアップウィンドウの上部に表示される文字列”,
  ”ポップアップの Text/Image path”
)
```

ポップアップ関数の引数の書き方は共通している. 上記にその記法を示す. ポップアップ関数の関数名には，表 3.6 のいずれかの関数名を指定する．第一引数は, popup 関数, popup_button 関数の場合は表示する文字列を, popup_image を使う場合は表示する画像へのパスを記述する. 第三引数には, 文字列と画像のパスのどちらかが指定可能である. 第三引数の値が画像のパスだった場合, ポップアップには画像が表示される. popup 関数の使用例を図 3.15 に示す. ここでは, popup_button 関数を使い, ポップアップウィンドウ内では画像を表示している.

navbar 関数のクエリ例

```
GENERATE ResponsiveHTML
  popup_button("Photo", "Image of a PC", "./img/pc.jpg")
FROM ...
```

図 3.15: popup 関数で生成されるポップアップの例

第 4 章

埋め込み型SuperSQL:処理系

本提案手法は，実現に際し，図 4.1 に述べる各機能の処理の実装を行った．SuperSQL 本体の実行用処理は PHP，SuperSQL 本体の処理は Java で記述されている．

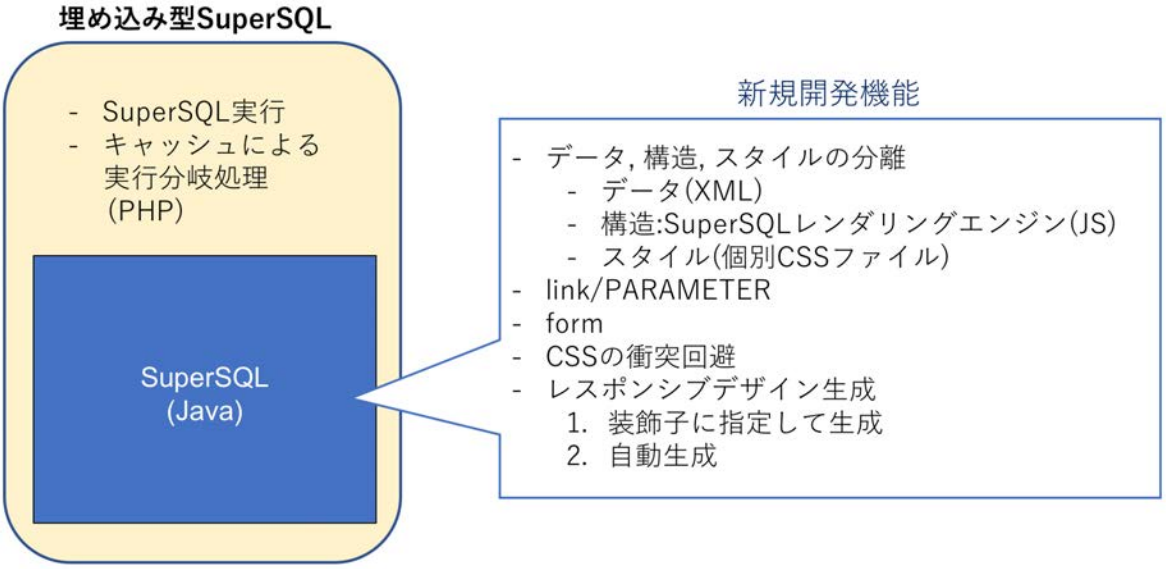


図 4.1: 埋め込み型 SuperSQL：全体

各機能の詳しい処理について以降の節で述べる．

4.1 実行関数

3.2 節で説明したように, SuperSQL を PHP 上で動作させるために, PHP の関数を実装した．ここでは, 実装した PHP の関数の具体的な内容について説明する．

4.1.1 ssql_setConfig 関数

ssql_setConfig 関数では SuperSQL の設定の読み込みを行う．引数には設定ファイル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）が可能となっている．ssql_setConfig 関数はページ内で 1 度だけ記述すれば, 同ページ内では設定情報が保持されるため, 同ページ内の複数のクエリを実行したい場合でも複数回記述する必要はない．ssql_setConfig 関数の内部では, 引数で受け取った値にしたがって以下のように SuperSQL の実行の際の引数へと変換する処理が行われている．SuperSQL の実行引数については, 2.4 節で述べたとおりである．

- ssql_setConfig() の引数: 設定ファイルのパス（config.ssql）
  - 変換後の SuperSQL の実行引数: -c [設定ファイルの相対パス]

  - 例: -c config.ssql

- ssql_setConfig() の引数: 設定項目の直接記述（データベースドライバ名やデータベース名等）
  - 変換後の SuperSQL の実行引数: -driver [ドライバ名] -db [データベース名] -h [ホスト名] -u [ユーザ名] ...
  - 例: -driver postgresql -db masato -h localhost -u masato ...

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．

Algorithm 1

```
1: switch ssql_setConfig() の入力引数 do
2:     case 引数が NULL return Error
3:     case 引数が一つ return -c [第一引数の値]
4:     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]
```

4.1.2 ssql_exec 関数

ssql_exec 関数では 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閲覧者によるページ訪問時に PHP が実行されるのと同じタイミングで SuperSQL が実行されるため, 動的な Web コンテンツの埋め込みが可能となった.

提案手法の実現にあたり, 新たに 3 つの SuperSQL の実行引数を追加した. 以下にそれを示す.

- -query ‘[SuperSQL クエリ]’
  - 指定したクエリを実行
  - 例: -query ‘GENERATE HTML <TFE> FROM $r_1, r_2, ..., r_n$ 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 or 簡易実行（4.5.3 項, 4.5.4 項参照）

- -querynum [ページ内に存在する SuperSQL クエリの連番]
  - ページ内のクエリの連番の設定
  - 例: -querynum 1 → ページ内 1 個目の SuperSQL クエリであることを示す.

- -ehtmlarg [{ 値 1, 値 2, ⋯}]
  - 受け渡す値の設定（4.2.2 項参照）
  - 例: -ehtmlarg {1} → 1 を SuperSQL へ受け渡すことを示す.

従来の SuperSQL では主に-f オプションを使用し, クエリファイルを指定して 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, 生のクエリを実行するため, 上記のような-query オプションの実装を行った. また, 提案手法の処理系は, 従来の GENERATE HTML のものと共通の部分が多いため, 同じメディア（HTML）の指定を行っている. そこで, 従来の SuperSQL の GENERATE HTML と提案手法の実行の処理系の分岐を行うために-ehtml オプションと-incremental オプションの追加を行った（4.5.3 項, 4.5.4 項）. -querynum オプションは, 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. この実行オプションは 4.5 節で主に利用する.

ssql_exec 関数の内部では, ssql_setConfig 関数によって変換された SuperSQL の実行引数に加え, -query オプション, -ehtml or -incremental オプション, -o オプション, -d オプション等の追加を行い, PHP の shell_exec 関数を使用して SuperSQL の実行を行うシェルスクリプトファイル（ssql.sh）の実行処理を行っている. 本関数の引数を入力とした場合の処理アルゴリズムを Algorithm 2 に，PHP 関数部の一連の処理の流れを図 4.2 に示す.

本関数の第二, 第三引数に指定可能なキャッシュ等の処理については，4.5 節で述べる.

4.1.3 埋め込み型 SuperSQL の返り値

提案手法は, 従来の SuperSQL の Web ページ生成とは異なり, Web コンテンツを構築するファイル郡の生成, コピー（4.4.1 参照）の他に Web コンテンツを載せる基盤となる HTML コードや既存コードのヘッダー内へ書き込みを行う JavaScript コード等（4.4.2 項参照）を返り値としている. 以下の「基盤 HTML コード」に返り値となる HTML コードを示す.

基盤 HTML コード

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
```

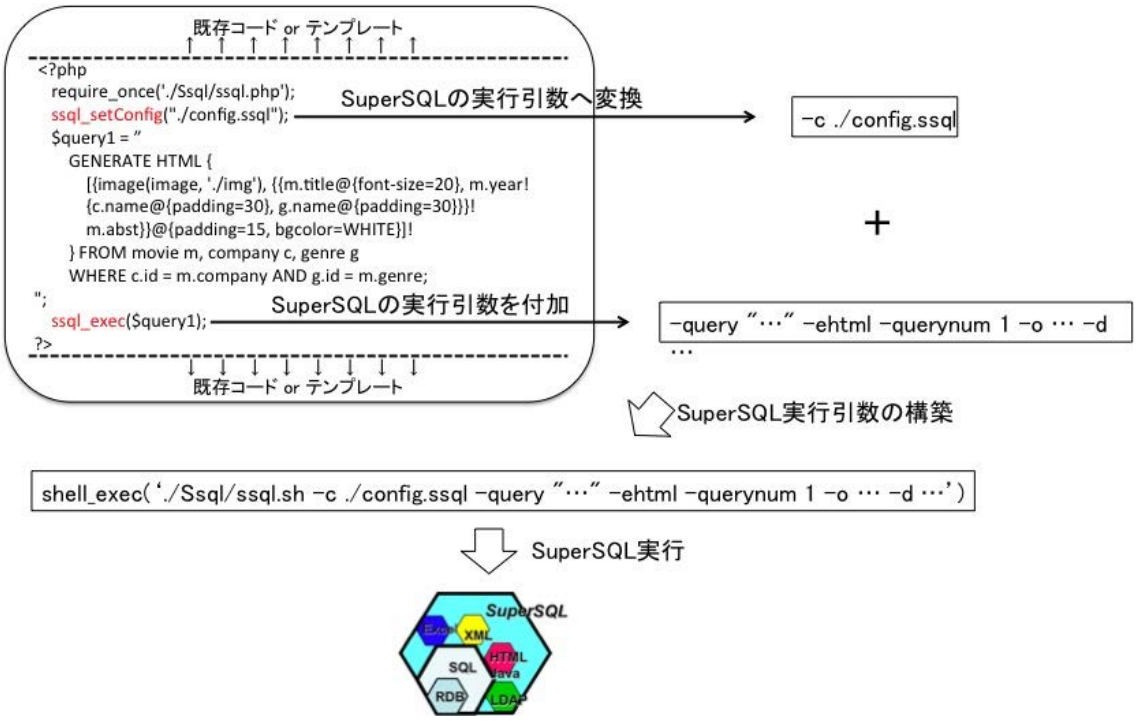


図 4.2: 埋め込み型 SuperSQL の処理の流れ

```
1 /**
2  * Webコンテンツを載せる基盤HTMLコード
3  */
4 <div id="ssqlResult1">
5 </div>
```

div タグで指定している id は‘ssqlResult’に-querynum オプションで指定された値を付加して生成される. 最終的に構築された Web コンテンツは上記の基盤 HTML コードにアペンドされる.

4.2 パラメータ受け渡し機構

4.2.1 plink, glink 関数と parameter 句の処理

従来の link 関数と foreach 句の処理を図 4.3 に, 新しい plink 関数, glink 関数と parameter 句の処理を図 4.4 に示す. link 関数, foreach 句は, 開発者のクライアントサイドで予め SuperSQL を実行し, 静的な Web ページを生成する. link 関数で生成される Web ページは, 引数で指定したクエリファイル (図 4.3 の foreach.ssql) が生成する HTML へのリンクが作成される. foreach 句では, link 関数を用いたファイルのリンク先となるファイル群を静的にリンクの数だけ生成される.

一方で, 提案手法の plink, glink 関数, parameter 句では, 閲覧者がページを訪問した際に PHP が実行される特定のタイミングでサーバサイドで SuperSQL が実行され, 結果が返されるため引数で指定したファイルへのリンクが動的に生成される. このとき, plink 関数を用いていた場合は POST 形式で値の送信を行い, glink 関数を用いていた場合には GET 形式で値の受け渡しが行われる.

続いて, 閲覧者がページ内のリンクをクリックして, parameter 句を使用したクエリが埋め込んであるページ（図 4.4 の parameter.php）へアクセスした際にも, サーバサイドで SuperSQL が実行され, リンク元のページから受け取った値をもとに動的に Web コンテンツが生成される.

4.2.2 SuperSQL への値の受け渡し

図 4.4 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．SuperSQL への値の受け渡しは, SuperSQL の新たな実行引数として 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{

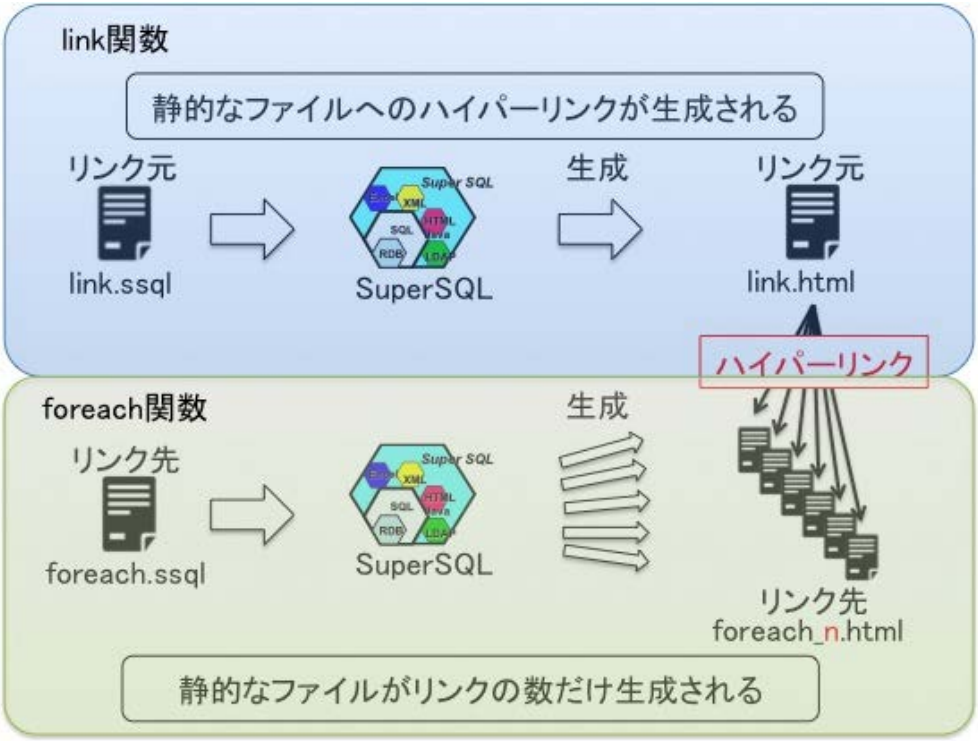


図 4.3: link 関数と foreach 句の処理

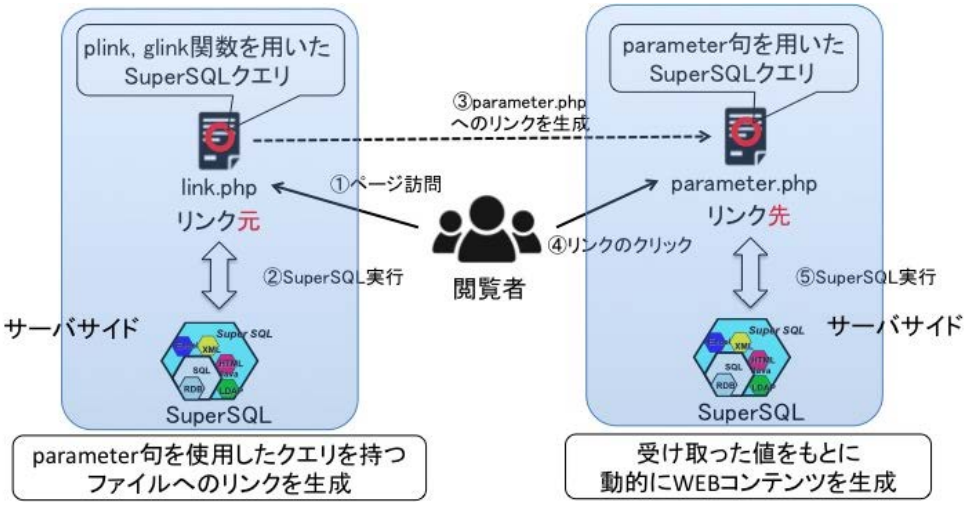


図 4.4: plink, glink 関数と parameter 句の処理

値 1, 値 2, …} といった形で指定して SuperSQL へ受け渡す．本処理のアルゴリズムは下記のとおりである．

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
```

上記のアルゴリズム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い合わせを実行している．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 問い合わせを行う直前に 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'
```

最終的に上記のクエリへと書き換えられ, SQL に問い合わせが行われる.

4.3 入力 (form) の処理機構

4.3.1 入力 (form)：手続き的実装 vs 宣言的実装

入力 (form) の手続き的実装では，一般に開発者が登録 form 用の HTML，入力値チェック用の JavaScript，SQL インジェクション対策を含む登録・更新・削除の処理が書かれた PHP を記述する．対して，提案手法による宣言的な実装では，開発者が一つの SuperSQL クエリを書くと，それを基にして前出の 3 つのコードが生成される．この違いについて，図 4.5 に示す.

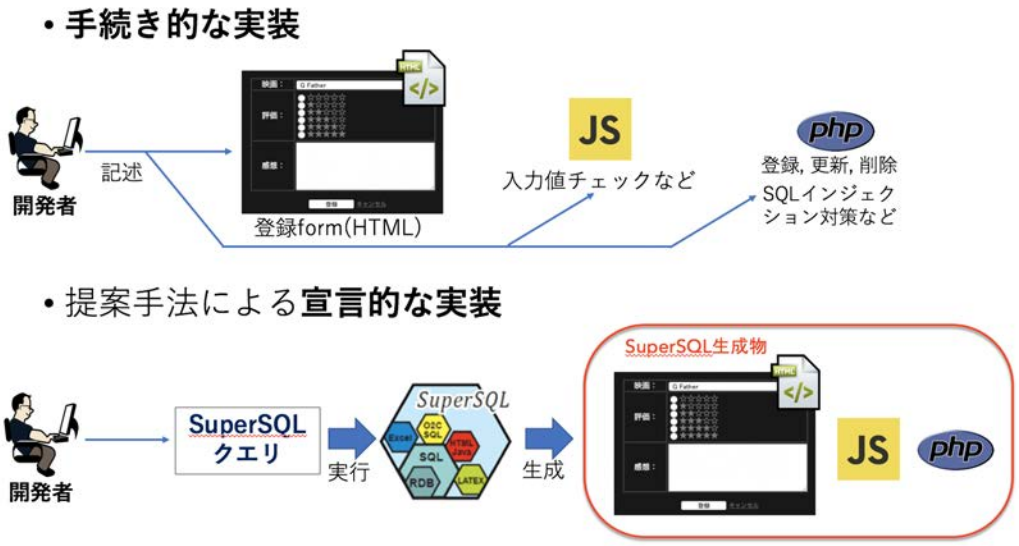


図 4.5: 入力 (form)：手続き的実装 vs 宣言的実装

4.3.2 入力 (form)：処理機構

前章で説明した提案手法の form 機能では，入力値や装飾子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 3.3.3 項の表 3.3 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．そしてその SQL により取得された 2 カラムの値をそれぞれ，第一属性 (第一カラム) の値を form の選択肢として，第二属性 (第二カラム) の値をデータベース格納時の値として使用する．第二属性が未指定の場合，データベースには第一属性の値が格納される．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}}
}@{insert}
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 TFE 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，取得された値の第一属性 e.name の値が form の選択肢として，第二属性 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．また，第三属性以降に何らかの属性名が記述されていた場合，実行は正常に行われるが，その属性により取得された値は本処理では使用されない．

本処理では，フロントエンドの form 用 HTML コードだけではなく，Web ユーザによる form 登録時の値のバリデーション用の JavaScript コードや，データベース格納処理のための SQL を含んだ PHP コードも併せて生成される．

4.4 クライアントサイド処理機構

クライアントサイド処理機構の詳細を下記に示す．なお, 本節内の図と例で使用している映画の画像と解説は, Yahoo!映画から引用している[1]．

4.4.1 クライアントサイドでの Web コンテンツの構築

近年, CSS でスタイルを分けて管理する他にも XML や json データを読み込み, JavaScript や jQuery によってクライアントサイドで Web コンテンツを構築する手法も見られる．このようなクライアントサイドでの Web コンテンツの構築による大きなメリットは, メンテナンス性の向上である．また, 最近では Web のデータを二次利用したいという需要が増えている．そういった需要に対しても, XML や json でデータを管理しているため, これらのファイルをダウンロードするだけで応えることができる．クライアントサイドにかかる処理の負荷についても, 近年のブラウザは JavaScript 等のクライアントサイドで動作する言語を高速に処理できるようになってきているため, そこまで大きな問題とはならない．

そこで提案手法では, スタイルの分離を行い, CSS を用いてクライアントサイドでスタイルのレンダリングを行うだけではなく, XML と JavaScript を用いてクライアントサイドでレンダリングを行い Web コンテンツを構築する機構を実装した．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 4.6 に示す．

また，上記で述べた分離生成される XML, JavaScript, CSS について下記に示す．このファイル分離により，4.5 節で述べるキャッシュを用いた処理に際して，必要最小限のファイルのみを生成することが可能となる．

データと構造（XML）

データと構造の役割を担う XML について説明する．提案手法で生成する XML ファイルは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

[1] 出典：https://movies.yahoo.co.jp/

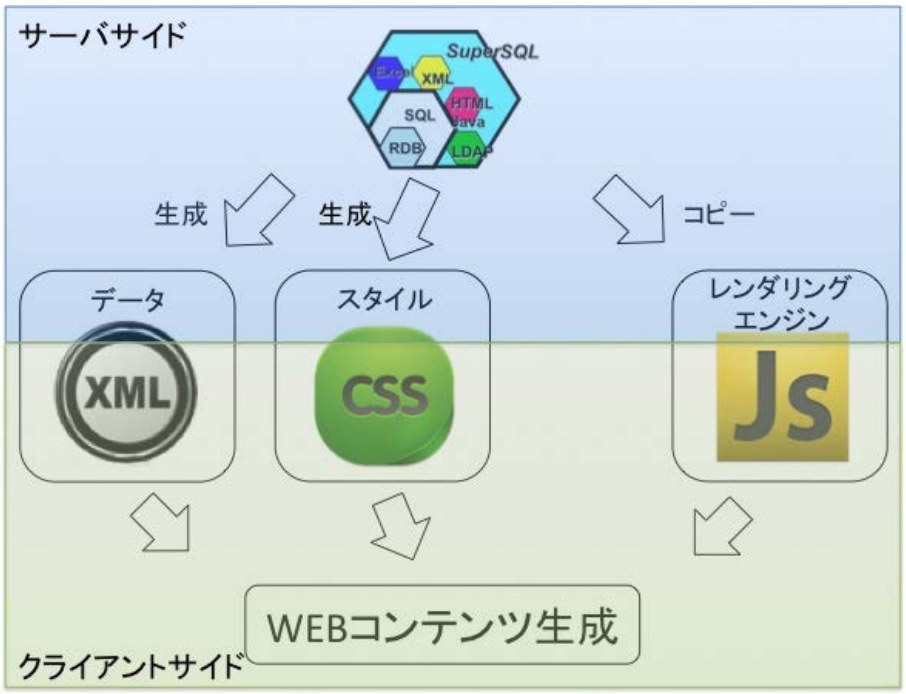


図 4.6: クライアントサイドでの Web コンテンツ構築の流れ

- データレイアウトの出力形式（div タグ, table タグ）
- TFE の ID 情報

また, XML は入れ子の構造をとっている. 図 4.7 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.

要素名は結合子であれば Connector, 反復子であれば Grouper, 値であれば Value, 画像であれば img とし, ひと目でわかる構造となっている. また, 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 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. C1, C2, G1, G2 はそれぞれ 2.2.2 項と 2.2.3 項で述べた略記法に従っている.

なお, XML ファイルは提案手法の実行ファイルが格納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’ として生成している.

SuperSQL レンダリングエンジン（JavaScript）

XML を解析し, Web コンテンツを構築する役割を担う, 提案手法における実質的なコー

SuperSQLクエリ

```
GENERATE HTML {
  [ { { { m.title||' ('||m.year||')'}@{ font-size=24, padding=12 }!
  { image( image, './img' ), {'作品紹介'@{ padding=15 } !
  m.abst@{ font-size=18, line-height=160 } } } !
  'レビュー'@{ padding=15 } !
  [ image( rank || '.png', './img' )@{ width=150, height=30 } ,
  r.comment@{ padding-left=15 } ]! }@{bgcolor=#333333} } ]!
} FROM movie m, review r
WHERE m.id = r.m_id;
```

生成されたWEBコンテンツ


```
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、ぜんそくの療養を目的に親戚が生
               活している海沿いの村にやって来た。そんなある日、彼女の前に誰もいない屋敷の青い窓に閉じ込められた、き
               れいなブロンドの少女マーニーが姿を見せる。その出会い以来、杏奈の身の回りでは立て続けに奇妙な出来事
               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、ジブリで期待してた分、残念
               </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 4.7: 生成される XML ファイルの例

ドジェネレータに相当する SuperSQL レンダリングエンジン（JavaScript）について説明する．まず，本レンダリングエンジンの処理アルゴリズムの一部を以下に示す．入力は 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全ての 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．

この SuperSQL レンダリングエンジンは実行する SuperSQL クエリの中身に関係なく共通の JavaScript ファイルである．XML ファイルや CSS ファイルは実行する SuperSQL クエリの内容によって中身が異なる一方で, この JavaScript ファイルは変化しないため, SuperSQL が実行された際に予め用意しておいた JavaScript ファイル（makeContents.js）を提案手法の実行ファイルが格納されているディレクトリにコピーして利用する．

makeContents.js と生成した XML を既存ページ内から読み込ませるためには, 既存ページのヘッダー内に書き込みを行う必要があるため, 4.4.2 項と同様の処理を行うことで実現した．以下が既存ページのヘッダー内に makeContents.js と生成した XML の読み込みの追加を行うコードである．

```
1 /**
2  * 既存ページのヘッダー内への書き込み
3  */
4 <script type=”text/javascript”>
5   // XMLファイルのパスの配列をグローバル変数で定義
6   var ssqlXmlFile = [];
```

```
7    ssqlXmlFile[0] = "既存ファイルのパス/GeneratedXML/既存ファイル名/ssqlResult1.xml";
8
9    // jQueryライブラリの読み込み
10   var jq = document.createElement("script");
11   jq.setAttribute("src","jscss/jquery.js");
12   document.getElementsByTagName("head")[0].appendChild(jq);
13
14   // makeContents.jsの読み込み
15   var make_contents = document.createElement("script");
16   make_contents.setAttribute("src","jscss/makeContents.js");
17   make_contents.setAttribute("type","text/javascript");
18   document.getElementsByTagName("head")[0].appendChild(make_contents);
19 </script>
```

次に, makeContents.js の XML の読み込みを行っている部分（一部）のソースを以下に示す.

```
1  // 構築されたWebコンテンツを載せる基盤の配列
2  var base = [];
3
4  // 構築されたWebコンテンツのID
5  var contentID = [];
6
7  var num = 0;
8  var root;
9
10 // 解析するXMLファイルのパスを格納する配列
11 var xmlpath = [];
12
13 // ページ内で実行されたSuperSQLクエリの個数分回す
14 for(var i = 0; i < xmlFileName.length; i++){
15   xmlpath[i] = "."+ssqlXmlFile[i].slice(ssqlXmlFile[i].search("GeneratedXML")-1);
16   contentID[i] = "ssqlResult" + (i+1);
17 }
18
19 window.onload = function(){
20   readXML();
21 }
22
23 /**
24  * XMLファイルの読み込み
25  */
26 var readXML = function(){
27   base[num] = document.getElementById(contentID[num]);
28   $.ajax({
29     async: true,
30     dataType: 'xml',
31     url: xmlpath[num],
32     success: function(data) {
33       traverseXML(data.documentElement);
34       base[num].appendChild(root);
35     }, complete: function(data){
36       num+=1;
37       // ページ内で実行されたSuperSQLクエリの個数分再帰的に実行
38       if(num < divID.length){
39         readXML();
40       }
41     }
42     });
43 }
```

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 makeContents.js を用いて以下の 1〜3 の処理を 1 から順番に，ページ内のクエリの数だけ再帰的に行っている.

1. XML ファイルの読み込み（readXML()）
2. XML ファイルの解析, Web コンテンツの構築（traverseXML()）
3. Web コンテンツを載せる基盤（base[num]）にアペンド

XML を読み込み, XMLDOM を形成する処理には, jQuery の$.ajax を使用している.

次に, 実際に XMLDOM を解析して Web コンテンツの構築を行っている makeContents.js の一部を以下に示す.

```
1  /**
2   * XMLDOMを解析し, Webコンテンツを構築
3   */
4  var traverseXML = function (node) {
5    // 引数のノードの要素名をチェック
6    if ((node.tagName.indexOf('Grouper') !== -1) || (node.tagName.indexOf('Connector')
       !== -1)) {
7      // 結合子, 反復子の種類によって処理の分岐
8      switch (node.getAttribute('type')) {
9        case 'G1':
10         // レイアウトタイプのチェック
11         if (node.getAttribute('outType') == 'div') {
12           // divレイアウトでのWebコンテンツの構築
13           var div = document.createElement('div');
14           // classの指定
15           div.classList.add('row');
16           div.classList.add(node.getAttribute('class'));
17           initProcess(div);
18           setArg(arguments[1], div);
19           var i = 0;
20           while (i < node.children.length) {
21             // 子ノードの数だけ再帰的に探索
22             traverseXML(node.children.item(i), div);
23             i=(i+1)|0;
24           }
25         } else if (node.getAttribute('outType') == 'table') {
26           // tableレイアウトでのWebコンテンツの構築
27           ...
28         }
29         break;
30       case 'G2':
31         ...
32       case 'C1':
33         ...
34       case 'C2':
35         ...
36     }
37   } else if (node.tagName == 'Value') {
38     // レイアウトタイプのチェック
39     if (node.getAttribute('outType') == 'div') {
40       // divレイアウトでのWebコンテンツの構築
41       var div = document.createElement('div');
42       div.appendChild(document.createTextNode(node.textContent));
```

```
43          // classの指定
44          div.classList.add(node.tagName);
45          if (arguments[1]) {
46            arguments[1].appendChild(div);
47          }
48        } else if (node.getAttribute('outType') == 'table') {
49          // tableレイアウトでのWebコンテンツの構築
50          ...
51        }
52      }
53  }
```

traverseXML() では, 子ノードを持つ限り再帰的に XMLDOM の探索を行い, Web コンテンツの構築を行っている. 上記のコードではソースコードの量の都合上, 反復子 G1（[],）以外の反復子, 結合子での処理, table レイアウトでの Web コンテンツの構築, SuperSQL の関数等の処理, その他を省いて載せている. 構築の完了した Web コンテンツは最終的に基盤（base[num]）にアペンドされ, 提案手法を使用している既存ページへ返される.

スタイル（CSS）

スタイルの HTML からの分離を実現するために, 既存のスタイルの指定もタグ内に記述するのではなく, 独立した CSS ファイルとして管理する処理へと変更した. 図 4.8 に変更の一例を示す.

また，提案手法を実現するにあたって挙げられた大きな問題点の一つが, テンプレート等の既存コードに SuperSQL によって生成した Web コンテンツを埋め込むことによる, コード同士の衝突である. 従来の SuperSQL の処理系による Web コンテンツの生成, すなわちデータ一体型での Web コンテンツの生成では, 既存コードと SuperSQL によって生成した Web コンテンツのコードのスタイル (CSS) 部が互いに干渉しあってしまいうまく表示することができなかった. 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 CSS の ID を割り当て，既存コードに対する提案手法によるスタイル (CSS) の干渉が起きないようにした. ユニークな CSS の ID を割り当てることにより，提案手法で生成されているコードに対してのみスタイル (CSS) の適用が行われるため，既存コード内のスタイル (CSS) に対する干渉は起こらない形となっている.

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. 例えば，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.

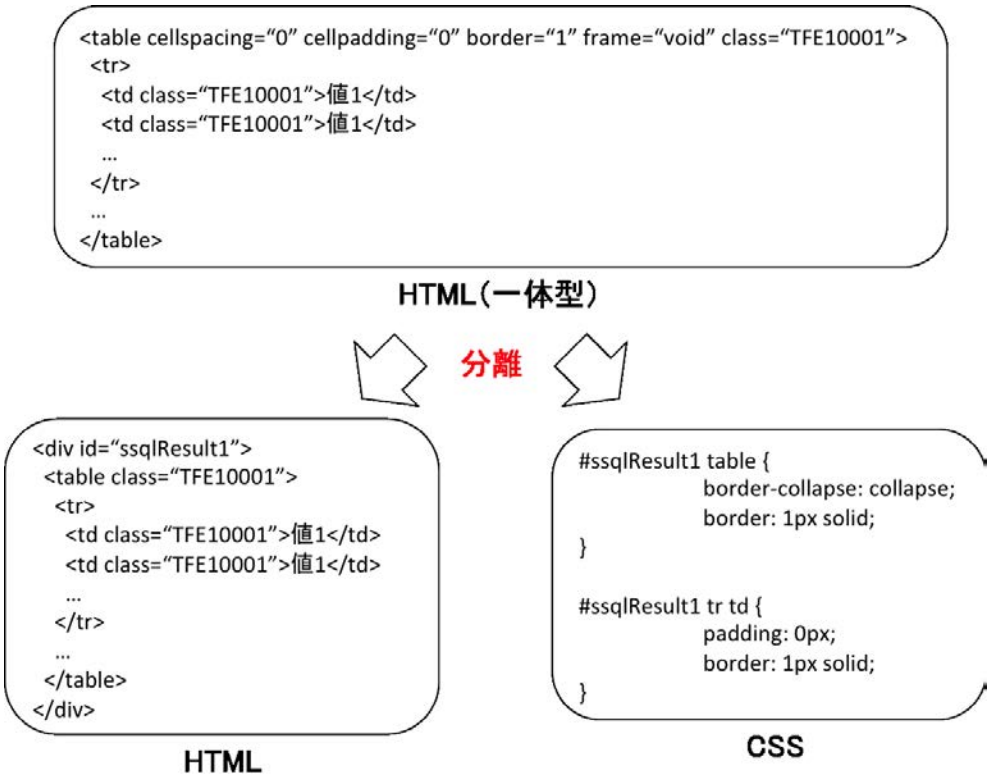


図 4.8: HTML と CSS の分離の例

4.4.2 ヘッダー情報の処理

従来の SuperSQL の GENERATE HTML では Web ページ全体を生成していたのに対し, 提案手法ではすでに存在している HTML ファイル等の BODY 内に SuperSQL で生成したコードを埋め込む．そのため，SuperSQL で生成する Web コンテンツに関するヘッダー情報を，提案手法を使用しているページの HEAD 部分に書き込みを行う必要がある．この処理を行うために, 提案手法では SuperSQL で生成したヘッダー情報を, JavaScript を用いて BODY 内から既存コードの HEAD 内に書き込みを行う実装を行った．以下にその一例を示す．

```
1  /**
2   * 既存ページのヘッダー内への書き込み
3   */
4  <script type="text/javascript">
5    var ssqlCSS[];
6    ssqlCSS[0] = document.createElement("link");
7    ssqlCSS[0].setAttribute("rel", "stylesheet");
8    ssqlCSS[0].setAttribute("type", "text/css");
9    ssqlCSS[0].setAttribute("href", "jscss//index/ssqlResult1.css");
10   document.getElementsByTagName("head")[0].appendChild(ssqlCSS[0]);
11 </script>
```

4.4.3 div タグでの Web コンテンツの生成

従来の SuperSQL の GENERATE HTML では table タグでの Web コンテンツの生成を行っていたが, 提案手法ではデフォルトでは div タグでの Web コンテンツの生成を行っている（図 4.9）.

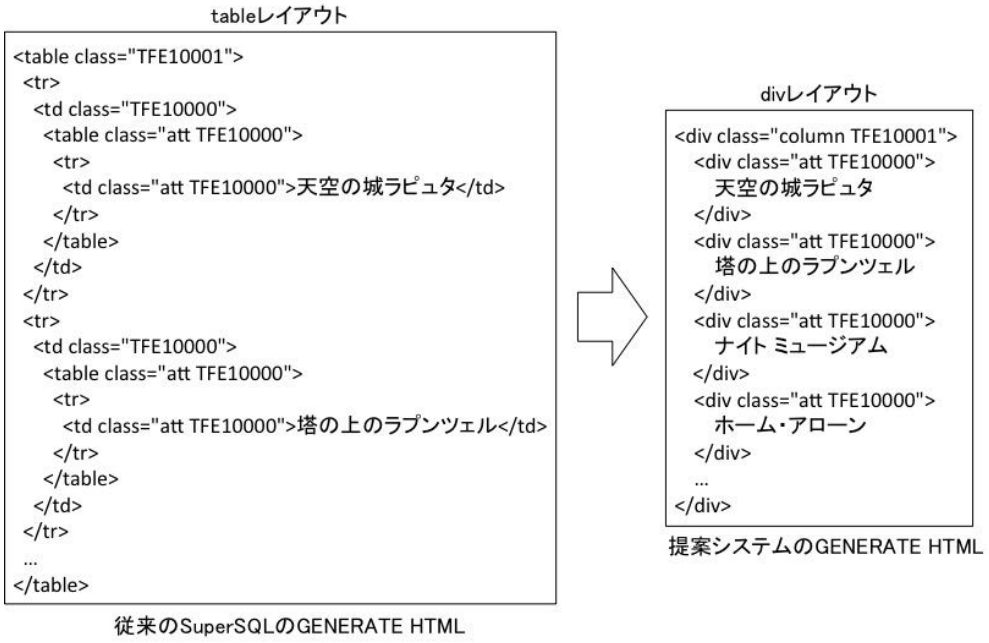


図 4.9: デフォルトのレイアウトに div を使用

近年の Web では, div タグと CSS を用いた div レイアウトが主流となっており, レイアウトの自由度が極めて高く, スタイル部を CSS として分離して管理することにより, ソース自体も見やすくまとまり, メンテナンス性も向上するからである. また, 反復子を複数使用し, 入れ子のクエリを記述した際に table タグだと構造が複雑になってしまい, 可読性が悪くなってしまうということが考えられるが, div タグを使用すると構造はわかりやすくなる.

従来の table レイアウトでは td タグ, tr タグによって SuperSQL の結合子, 反復子の水平結合（,）, 垂直結合（!）による結合を実現していたが, div レイアウトでの Web コンテンツの生成を行うにあたり, flex-direction CSS プロパティによってこれらの結合を実現した. div レイアウトにおける水平結合（,）, 垂直結合（!）に対応したスタイルの記述を表 4.1 に示す.

表 4.1 のスタイルの生成を行うことで, div タグ内の class に row を指定すると横結合を, column を指定すると縦結合を実現することが可能なった. 図 4.10 に簡単な div レイアウトにおける Web コンテンツの例を示す.

もちろん, table タグを用いた table レイアウトが適している場合もあるため, 装飾

表 4.1: div レイアウトにおける水平結合（,）, 垂直結合（!）に対応したスタイルの記述

結合方向	table レイアウト	div レイアウト（CSS の記述）
水平結合（, [],）	td タグ	.row { display: flex; flex-direction: row; }
垂直結合（! []!）	tr タグ	.col { display: flex; flex-direction: column; }

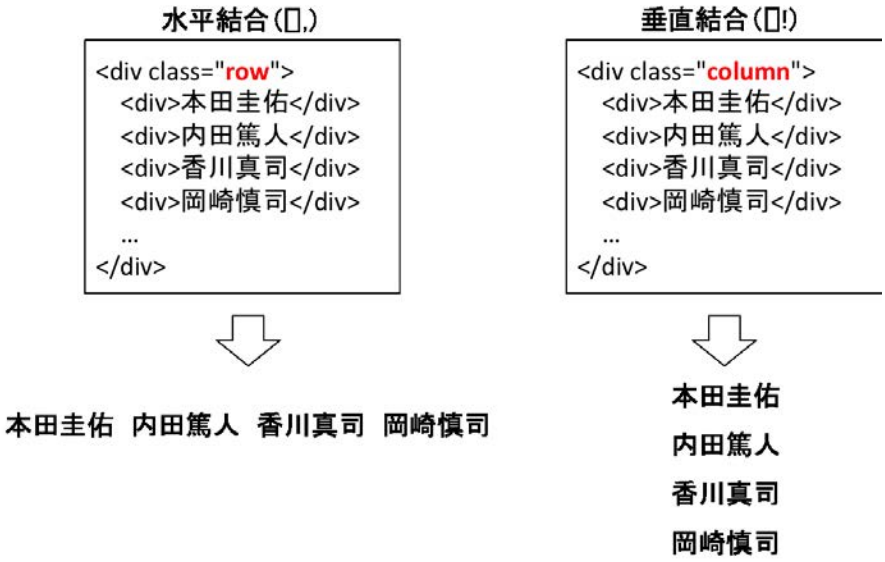


図 4.10: div レイアウトにおける水平結合と垂直結合

子@{table} を用いることで table タグでの Web コンテンツの生成も可能である. 装飾子@{table} で囲った TFE 部分が table タグで生成される. さらにその内側で装飾子@{div} を使用することで, @{div} で囲われた TFE 部分を div タグで生成するといったことも可能である. 例を図 4.11 に示す.

図 4.11: 装飾子@{table} と@{div} の例

4.5 キャッシュを利用した実行の分岐

提案手法の SuperSQL の実行を担う PHP 部分の処理について，下記の実行方式を考案し検討を行った.

方式 1 従来の SuperSQL をそのまま実行

方式 2 従来の SuperSQL が実行される際，生成物（HTML）をキャッシュとして保存しておき，必要に応じて実行処理で使用

方式 3 従来の SuperSQL が実行される際，生成物を細かく HTML, XML, JavaScript, CSS の形に分離してキャッシュとして保存しておき，それぞれ必要に応じて実行処理で使用

提案手法では, 従来の SuperSQL の GENERATE HTML の静的な Web ページを生成とは異なり, 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, 方式 1 を利用した場合，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, 方式 2 を利用した場合，例えばデータベースの値が更新されてデータのみが変わった場合にもスタイル等を含む HTML コード全体が生成されてしまい, 処理の無駄であると考えられる.

これらを踏まえ, 提案手法では方式 3 を実行方式として採用した．本方式では，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 等はキャッシュされたものを用いる．また，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 もキャッシュされたものを使用する機構を実現した．この提案手法の SuperSQL 本体内で分岐される実行方法について以下に示す.

- 通常の実行
  - 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
- 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 (データ更新なし)
  - 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分岐のコントロールフローを図 4.12 に示す. 設定等の読み込みから結果出力の前までの処理はサーバーサイドで行われる.

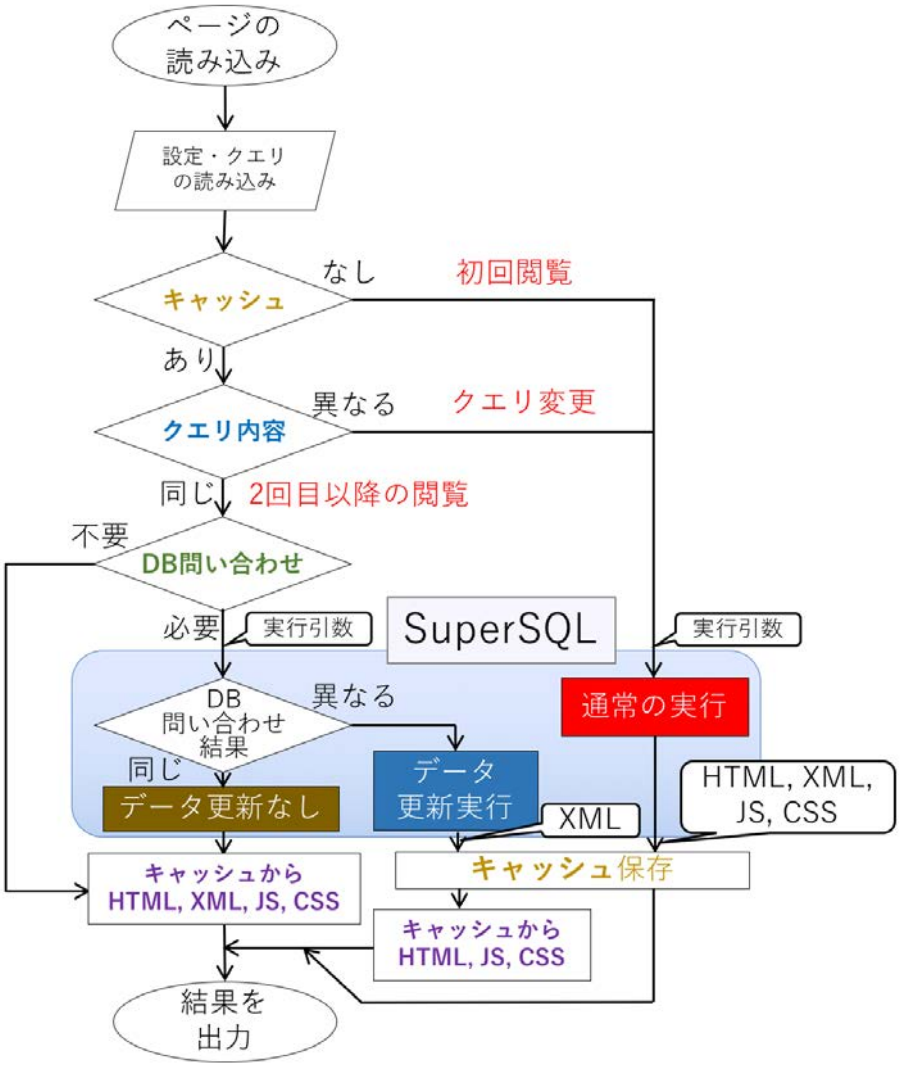


図 4.12: キャッシュを利用した実行分岐のコントロールフロー

図中のキャッシュの保持時間は 3.1 節で述べた ssql_exec 関数の第二引数，もしくは ssql_cacheTime 関数，データベースへの問い合わせ間隔については ssql_exec 関数の第

三引数，もしくは ssql_selectDB_interval 関数により指定可能であり，ともにデフォルトは 0 秒（キャッシュを使用しない, 常にデータベース問い合わせを行う）となっている．

また，既にキャッシュがありデータベース問い合わせも不要な場合は，上記の SuperSQL 本体内で行われる 3 通りの実行は行われず，簡易実行 (データ更新なし) のときと同様にキャッシュに保存されているファイルを用いて結果表示が行われる．

4.5.1 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 を使用している．このライブラリを使用している理由は, 軽量でありながら詳細な設定が可能であるからである．提案手法でこのライブラリを使用する上での設定は以下のようになっている．

```
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     'automaticSerialization' => 'true',
8     'lifeTime' => '86400'
9 );
```

‘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．‘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’ では配列を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, 有効にしている．‘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．上記の例では, この生存時間を 24 時間に設定している．すなわち, 1 日に一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．例えば, 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．

4.5.2 通常の実行

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 の処理系にも修正を加えた. 図 4.13 に埋め込み実行の 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い合わ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るため, 検索結果を MD5 で 128 ビットの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. 例えば, index.html というファイル内の 1 つ目の SuperSQL クエリの SQL の問い合わせ結果は, 'sqlResult/index/ssqlResult1.txt' として保存される. MD5 を用いているのは, 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することができるためである.

また, 検索結果が同じかどうかがわかればいいので, 比較するときも元に戻さずにハッシュ値同士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.

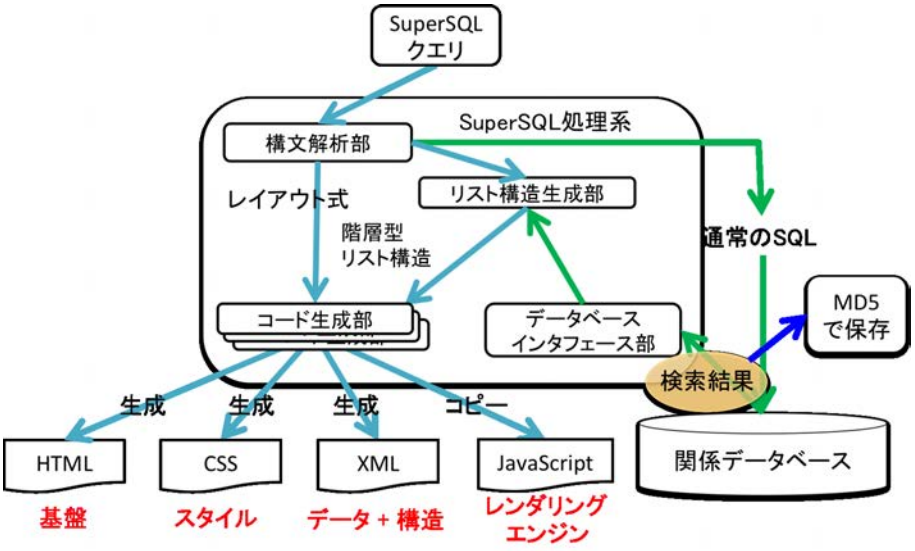


図 4.13: 通常の埋め込み実行の SuperSQL の内部処理

4.5.3 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 SuperSQL の処理の流れを図 4.14 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い合わせを行った際に, 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.

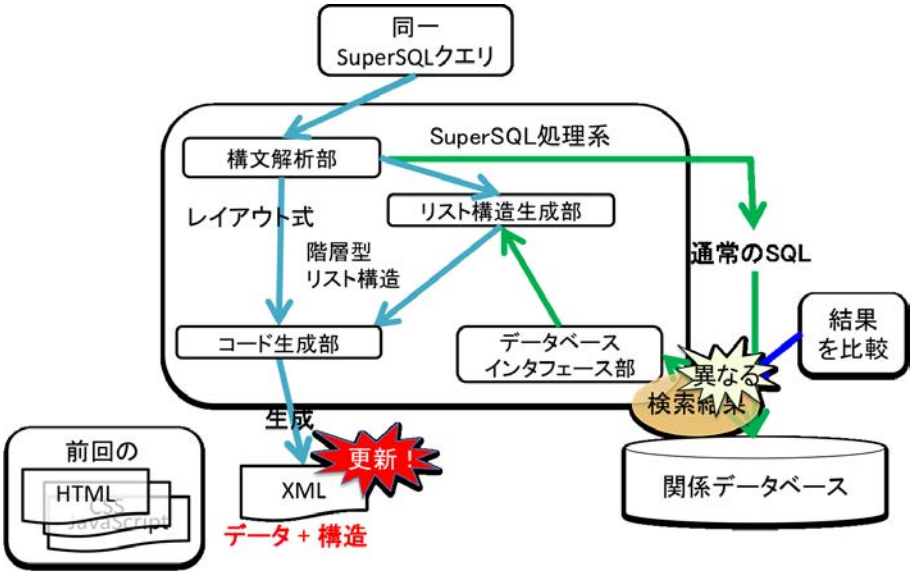


図 4.14: データ更新実行の SuperSQL の内部処理

4.5.4 簡易実行 (データ更新なし)

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 SuperSQL の処理の流れを図 4.15 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い合わせを行った際に, 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）と比較する. そして, 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になっている.

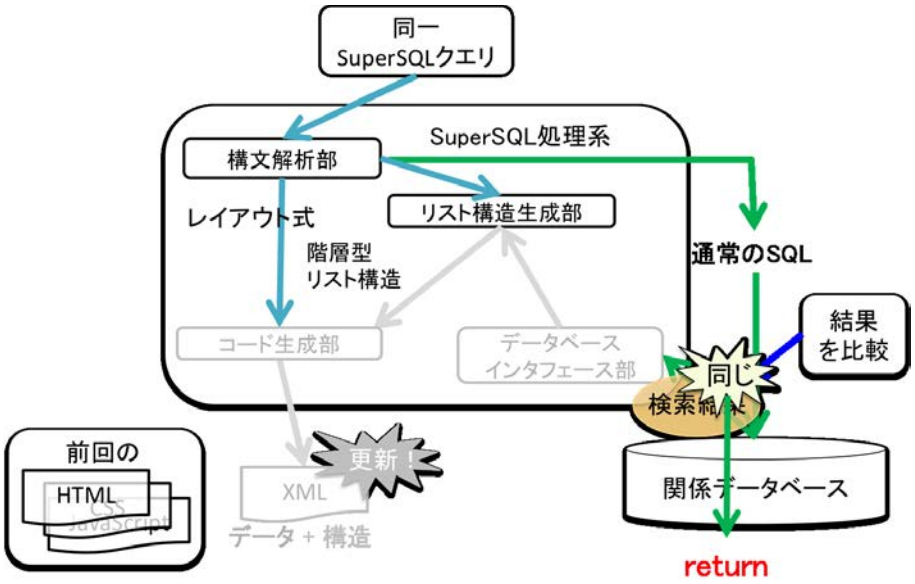


図 4.15: 簡易実行の SuperSQL の内部処理

4.5.5 キャッシュの保存先

提案手法では生成物を XML, JavaScript, CSS などといった形に細かく分離して生成する形をとっている．これらはキャッシュとしてサーバーサイドへ保存される．本提案手法では，XML 以外の JavaScript や CSS などの生成物は閲覧者ごとではなくページごとに共通なことが多く，またクライアント側への処理負荷も考慮して今回はサーバー側へキャッシュを保存する機構とした．

しかしながら，例えばログイン機能を有しているページを考えてみた場合，ログイン後のページ表示はユーザーごとに異なる可能性が高く，クライアント側へのキャッシュ保存も有用であると考えられる．こちらについては今後の課題である．

4.6 レスポンシブレイアウト生成機構

4.6.1 Bootstrap（世の中一般のレスポンシブデザイン実現手法）

本生成機構では, SuperSQL でレスポンシブデザインを実現する方法として, Web のフロントエンド開発用のフレームワークである Bootstrap の方法論を利用している．ここでは Bootstrap によってレスポンシブデザインを体系的に実現している方法について説明をする．Bootstrap は, CSS フレームワークの一種である．CSS フレームワークは, CSS の枠組みやスタイルなどをある程度最初から定義している, ライブラリファイルのようなもので, それぞれの CSS フレームワークが定義しているルールに沿って HTML

表 4.2: 各ブレイクポイントにおけるカラムの挙動

サイズ指定	xs	sm	md	lg	xl
ブレイクポイント	<544px	<768px	<992px	<1200px	≥ 1200px
想定している端末	スマートフォン	タブレット	ノート PC	デスクトップ PC	デスクトップ PC (大画面の PC)
グリッド上の挙動	常に横並びで表示	画面幅 ≥ ブレイクポイント のときは横並びで表示			
		画面幅 <ブレイクポイント のときは各要素の幅が 100%となり 縦並びで表示			

の class 名などを設定し既に用意されているスタイルを適用するだけで, 格段に速く見栄えを整えることが可能である.

Bootstrap では, コンテンツを配置していく考え方として, 可変グリッドレイアウトというものを採用している. これは, 画面を行 (row) と列 (column) から成る格子状に分割したグリッドと見立てて, そのグリッドに要素を配置し, 画面幅に合わせてコンテンツの幅や位置を変化させる方法である. 通常の Bootstrap では, 横幅を 12 等分にしたグリッドシステムが用意されており, それぞれの row において, 12 の振分けをいくつにするかで column を決めていく. 例えば 4:4:4 で等分の 3 カラムや, 8:4 の 2 カラムなどを作成し, その中にコンテンツを入れていく.

また, Bootstrap では, 画面幅に応じて, Web ページのレイアウトを動的に変化させる仕組みとして, 表示デバイスのウィンドウサイズによってレイアウトの表示を切り替えるポイントをブレイクポイント (breakpoint) と呼び, スマートフォン, タブレット, PC, 大画面の PC のサイズを想定した 5 つのサイズにおいて, どのように column を表示するかを指定するようにしている. それぞれの想定画面サイズに対するサイズ指定がどのような挙動になるのかをまとめた表を表 4.2 に示す.

実際に, row と column を使ったレイアウトを作成するには, HTML で div タグなどを用いたブロックレベルの要素に対し row の指定を行い, その中に column を作る場合は, ブレイクポイントと大きさを含む指定を行う. 例えば, ソースコード 4.1 に示す HTML では, 図 4.16 に示されるようなレイアウトになる. タブレットを想定した sm というブレイクポイントにおいて, それぞれ 4 と 8 というカラムサイズを指定している. この場合, 画面幅がブレイクポイントである 768px 以上の場合は, その割合が保たれたまま横並びで要素が配置され, 画面幅がブレイクポイントを下回った場合は, その割合の指定を放棄され, 要素が縦並びに配置される.

ソースコード 4.1: Bootstrap を使った HTML の例

```
1 <div class="row">
2   <div class="col-sm-4"> .col-sm-4 </div>
3   <div class="col-sm-8"> .col-sm-8 </div>
4 </div>
```

画面幅 ≥ 768px

.col-sm-4	.col-sm-8

画面幅 < 768px

.col-sm-4
.col-sm-8

図 4.16: Bootstrap を用いた HTML コーディングによる表示の例

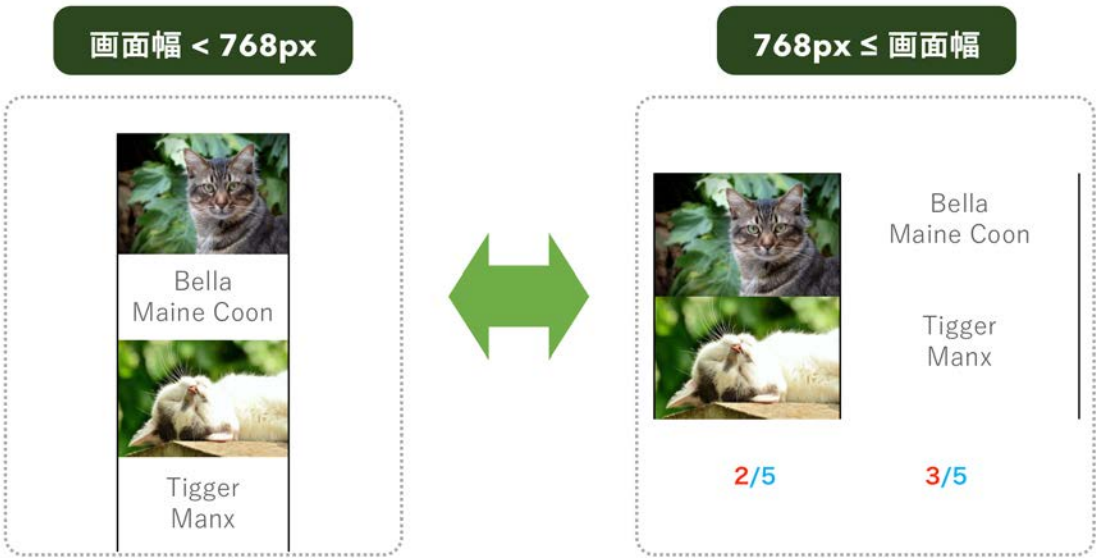


図 4.17: Bootstrap と Sass を用いたレスポンシブデザインの例

上記は最も簡単な例であるが，Bootstrap でより複雑なレスポンシブデザインを実装するためには，一般的に Sass と呼ばれる CSS プリプロセッサを併用する．Sass は従来の CSS では記述できなかった入れ子, 変数, 関数などの記述を可能にしている．使用時には Sass ファイルをコンパイルし CSS を作成する．例えば図 4.17 のような sm(768px) をブレイクポイントとするレスポンシブデザインを実装する場合，Bootstrap のコードを含む HTML と Sass のコードは図 4.18，図 4.19 のような記述となる．

こちらは簡単な例であり，複雑なレスポンシブデザインを実装しようとするほど HTML と Sass(CSS) のコードもより複雑になり，どのようなレスポンシブデザインになっているのかコードを一見しただけでは判断がつかなくなる．一般的な Bootstrap

HTML

```
<div class="TFE10004">
  <div class="row">
    <div class="TFE10000">
        <img src="../images/bella.jpg">
    </div>
    <div class="TFE10003">
        <div class="row">
            <div class="TFE10001">Bella</div>
        </div>
        <div class="row">
            <div class="TFE10002">Maine Coon</div>
        </div>
    </div>
  </div>
    :
```

図 4.18: Bootstrap の記述を含む HTML コードの例

CSS (Sass)

```
.TFE10004{
    $grid-columns:5 !global;
    .TFE10000{
        @include make-sm-column(2);
    }
    .TFE10003{
        @include make-sm-column(3);
    }
}
```

図 4.19: Bootstrap の記述を含む Sass コードの例

では実装したいレスポンシブデザインが複雑になるとコードが分かりにくくなるが，提案手法による宣言的な記述ではそうはならない．次項では，提案手法について説明する．

4.6.2 SuperSQL におけるレスポンシブレイアウト生成

提案機構では, SuperSQL を使って Web ページ生成を行うために，コード生成部（Parser）に新たな機構を実装した．これにより, 4.6.1 項で説明した Bootstrap のレスポンシブデザインの方法論に沿った形の HTML, CSS, Javascript を生成可能とした. 本研究で実装したレスポンシブな Web ページ生成機能は, デスクトップ向けの Web ページを生成する機能 (GENERATE HTML) や, モバイル向けの Web アプリケーションを生成する機能 (GENERATE Mobile_HTML5) とは, レイアウトの生成方法が根本的に異なるため, メディア（出力媒体）として「ResponsiveHTML」を新たに実装した. GENERATE ResponsiveHTML では, 従来の SuperSQL に対して以下の点を変更した.

- 出力される Web コンテンツが row と column によって構成されるグリッドに配置されるよう HTML の出力機構を実装
- 出力される Web コンテンツの内容に合わせて, 柔軟にレスポンシブなグリッドを CSS で生成するために, Sass を出力, コンパイルする機構を導入
- 5 段階の画面サイズに対応したコンテンツの大きさが指定できるように新たな装飾子を実装

HTML 出力機構　従来の手法における HTML のレイアウト出力は, GENERATE HTML では table タグを使用し, GENERATE Mobile_HTML5 では div タグを利用していた. table タグは, 表の出力を行うための HTML の機能なので, 適用できる CSS の装飾などが限られており, 幅広い Web コンテンツの格納には向いていない. このような理由から, GENERATE Mobile_HTML5 ではレイアウト作成を div タグで行う方法へと移行し, スマートフォン用の UI に特化したフレームワークである jQuery Mobile を活用することでタッチ操作に最適化された Web ページの生成を実現した. 提案手法の GENERATE ResponsiveHTML では, GENERATE Mobile_HTML5 と同じように div タグを用いてレイアウトを作成するが, より幅広い閲覧端末に対応可能なレイアウトを作成可能にするために, 行 (row) と列 (column) からなる可変グリッドの構造を HTML で組み立て, その中にコンテンツを配置していく方法をとっている.

実際の処理は，XML が生成され HTML としてレンダリングされる形となっているが，理解の容易性も加味し本節ではレンダリング後の HTML をベースに話を進める．

CSS 出力機構　生成される HTML に対応する CSS を生成する方法として, 提案手法では, CSS プリプロセッサの Sass を利用している. CSS と Javascript のパッケージとい

う形で提供されている通常の Bootstrap では, グリッドの最大カラム数が 12 に固定されており, あらかじめ定義された CSS によるレイアウトの作成には多くの制限が存在する. そこで, 提案手法では, Bootstrap のコアとなる CSS の分岐を多用したレイアウトの仕組みを Sass に落とし込むことによって, 主となるパラメータや機能を, 変数や, mixin（Sass においては，CSS のスタイルをまとめて定義することができ，引数に応じて一部の CSS スタイルを変更することができる関数のような機能）からアクセス可能にし, データ量や, クエリ内の指定に合わせて柔軟なレイアウトを生成可能にしている.

4.6.3 設計

レスポンシブ Web ページの生成機構の実現のために必要な HTML と CSS の出力機構について説明する.

SuperSQL では, 与えられたクエリの TFE を外側から内側の方向で再帰的に処理していく. 例えば,

```
GENERATE ResponsiveHTML
  [ att1 , { att2 ! att3 } ]!
FROM ...
```

上のようなクエリは, 構文解析部によって

```
[G2, [C1, [0], [C2, [1], [2]]]]
```

と表されるレイアウト式に変換され, 実行時にはレイアウト式の G2 → C1 → C2 の順番で処理される. レイアウト式内の, 数字の 0, 1, 2 はクエリ中の属性に対応するプレースホルダである (結合子と反復子の略記法は表 2.1 と表 2.2 を参照). しかし, 出力媒体が HTML などのマークアップ言語の場合, 開始タグで開いたタグは, 終了タグで閉じる必要があるため, 実際の HTML の出力の処理は, それぞれの属性値の出力も含んだ場合, G2 の開始 → C1 の開始 → att1 の出力 → C2 の開始 → att2 の出力 → att3 の出力 → C2 の終了 → C1 の終了 → G2 の終了といった流れになる.

また, SuperSQL 処理系のデータ構造生成部では, 構文解析部で生成されたツリー構造に合わせて, クエリ内の TFE の各要素に TFE ID といった番号付きの値が割り当てられる. TFE ID はそれぞれの要素を一意に識別することができる値なので, HTML の出力機構では, この値をレイアウトの各要素の HTML 出力の際に使用する div タグの class 属性の値として使用している.

また, SuperSQL の CSS の出力機構では, SuperSQL クエリ内の装飾子の設定に合わせて, 指定された要素の TFE ID を取得し, その TFE ID を持つ class 属性の要素に対し, スタイルを適用していく.

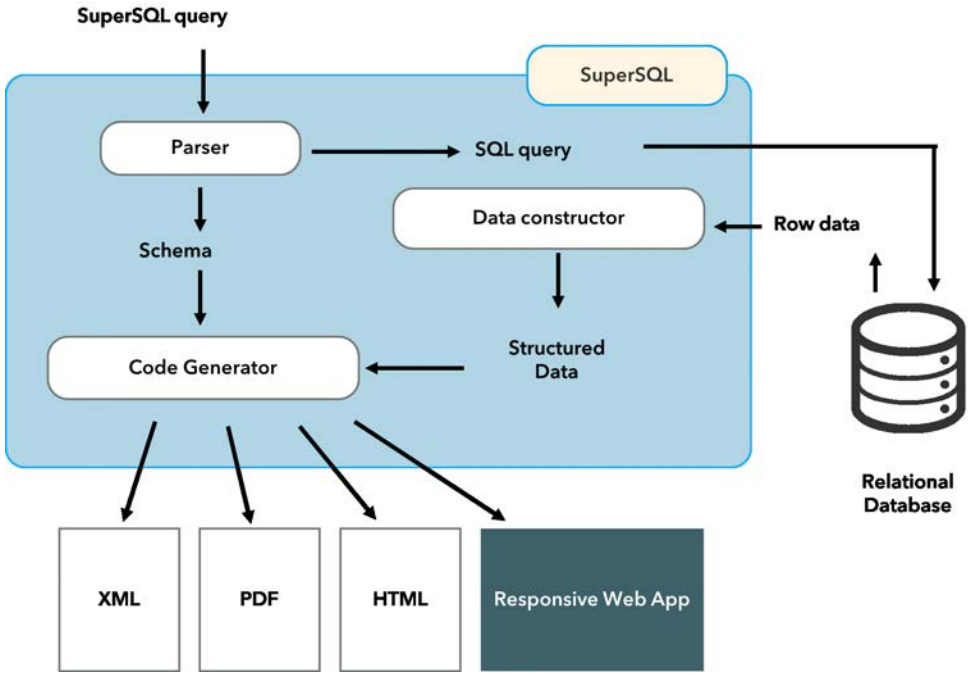


図 4.20: レスポンシブレイアウト生成機構を含んだ SuperSQL 処理系

従来の SuperSQL では, 装飾指定式と出力される CSS は非常にシンプルな対応関係を持っており, 装飾指定式で (項目名 = 値) として指定したものは, ほとんどの場合, 装飾指定式の「項目名」が CSS の「プロパティ」, 装飾指定式の「値」が CSS の「値」となる. 例えば, `@{font-size="15px"}`という風な装飾子の指定があった場合, 出力される CSS は, 指定先の HTML の class 属性の値が TFE100XX とした場合, `.TFE100XX{font-size: 15px;}`となる.

しかし, レスポンシブレイアウトを作成するための CSS は, 画面幅による条件分岐をするメディアクエリや, 幅広いブラウザで同じ挙動となるようにするための記述などが含まれるため, 従来の SuperSQL の装飾指定と出力 CSS のシンプルな対応関係では機能的に対応できない部分が多くなってしまう. こうした理由から, 提案手法では従来の CSS 出力機構に加え, Sass のコンパイラを SuperSQL の処理系に加えることで, Sass 経由での CSS の出力を可能とする. これにより, 変数と mixin(関数) を使って, レスポンシブレイアウトを実現するための複雑な CSS の出力を柔軟かつ効率的に行えるようにする. つまり, 従来の SuperSQL では, HTML では基本構造とデータコンテンツを出力し, CSS ではデザインの出力をしていたのに対し, 提案手法では HTML の出力処理で基本構造とデータコンテンツを, CSS の出力処理で細かいデザインを, Sass の出力処理でレスポンシブなレイアウトのための CSS の生成を行う.

4.6.4 指定生成：装飾子指定によるレスポンシブレイアウト生成

アーキテクチャ SuperSQL 処理系は, 構文解析部 (Parser), データ構造生成部 (Data Constructor), メディア生成部 (Code Generators) から成る. 今回の SuperSQL を用いたレスポンシブ Web アプリケーションの生成機構では主にコード生成部へ多くの処理を追加し, レスポンシブ Web アプリケーション向けの HTML, CSS, JavaScript, PHP の生成を行う機構を実装した (図 4.20).

内部処理 提案機構で実装した横結合, 縦結合, 横反復, 縦反復の HTML と Sass の出力アルゴリズムについて説明する. 横結合の処理アルゴリズムをアルゴリズム 7 に, 縦結合の処理アルゴリズムをアルゴリズム 8 に, 横反復の処理アルゴリズムをアルゴリズム 9 に, 縦反復の処理アルゴリズムをアルゴリズム 10 に示す. 共通しているのは, それぞれの構造演算子で結ばれた TFE 全体を表す TFE ID と, 構造演算子で結ばれる (又は含まれる) それぞれの要素の TFE ID を利用し, 可変グリッドにおける行 (row) と列 (column) の対応関係が適切に出力されるように HTML と Sass の出力処理を同時に行っている点である. 縦結合と縦反復では, シンプルに xs, sm, md, lg の装飾指定に合わせて Sass を生成していく横結合では, 3.3.4 で説明したように, 結合子で結ばれた要素にサイズ指定が一切ない場合, サイズ指定が一部ある場合, 全てにサイズ指定がある場合に合わせて, 各要素のサイズを決定する処理が入る. 横反復では, 同様に, 内部の要素にサイズ指定が無い場合は等分のサイズを設定, サイズ指定がある場合はそのサイズで出力するといった分岐の処理が行われる.

アルゴリズム 7 横結合 (C1) の処理

```
1: class_id ⇐ C1 の TFE ID
2: if 最上位の階層 then
3:    < div class="row" >を出力 (HTML)
4:    < div class="class_id" >を出力 (HTML)
5:    .class_id 用の column を生成 (Sass)
6: end if
7: < div class="row" >を出力 (HTML)
8: C1 内の要素のそれぞれのサイズを決定
9: while C1 内に未処理の属性がある間 do
10:   class_id2 ⇐ C1 内の要素の TFE ID
11:   < div class="class_id2" >を出力 (HTML)
12:   ..class_id2 用の column を生成 (Sass)
13:   次の階層の処理へ移動
14:   < /div >を出力 (HTML)
15: end while
16: < /div >を出力 (HTML)
17: if 最上位の階層 then
18:   < /div >を出力 (HTML)
19:   < /div >を出力 (HTML)
20: end if
```

アルゴリズム 8 縦結合 (C2) の処理

```
1: class_id ⇐ C2 の TFE ID
2: if 最上位の階層 then
3:    < div class="row" >を出力 (HTML)
4:    < div class="class_id" >を出力 (HTML)
5:    .class_id 用の column を生成 (Sass)
6: end if
7: while C2 内に未処理の属性がある間 do
8:    class_id2 ⇐ C2 内の各要素の TFE ID
9:    < div class="row" >を出力 (HTML)
10:   < div class="class_id2" >を出力 (HTML)
11:   .class_id2 用の column を生成 (Sass)
12:   次の階層の処理へ移動
13:   < /div >を出力 (HTML)
14:   < /div >を出力 (HTML)
15: end while
16: if 最上位の階層 then
17:   < /div >を出力 (HTML)
18:   < /div >を出力 (HTML)
19: end if
```

アルゴリズム 9 横反復 (G1) の処理

1: *class_id* ⇐ G1 の TFE ID
2: **if** 最上位の階層 **then**
3: < div class="row" >を出力 (HTML)
4: < div class="*class_id*" >を出力 (HTML)
5: .*class_id* 用の column を生成 (Sass)
6: **end if**
7: < div class="row" >を出力 (HTML)
8: **while** G1 内に未処理の属性がある間 **do**
9: *class_id*2 ⇐ G1 内の要素の TFE ID
10: < div class="*class_id*2" >を出力 (HTML)
11: **if** サイズ指定がある場合 **then**
12: サイズ指定を利用して .*class_id*2 用の column を生成 (Sass)
13: **else**
14: C2 内の要素数で等分したサイズで .*class_id*2 用の column を生成 (Sass)
15: **end if**
16: 次の階層の処理へ移動
17: < /div >を出力 (HTML)
18: **end while**
19: < /div >を出力 (HTML)
20: **if** 最上位の階層 **then**
21: < /div >を出力 (HTML)
22: < /div >を出力 (HTML)
23: **end if**

アルゴリズム 10 縦結合 (G2) の処理

```
1: class_id ⇐ G2 の TFE ID
2: if 最上位の階層 then
3:    < div class="row" >を出力 (HTML)
4:    < div class="class_id" >を出力 (HTML)
5:    .class_id 用の column を生成 (Sass)
6: end if
7: while G2 内に未処理の属性がある間 do
8:    class_id2 ⇐ G2 内の各要素の TFE ID
9:    < div class="row" >を出力 (HTML)
10:   < div class="class_id2" >を出力 (HTML)
11:   .class_id2 用の column を生成 (Sass)
12:   次の階層の処理へ移動
13:   < /div >を出力 (HTML)
14:   < /div >を出力 (HTML)
15: end while
16: if 最上位の階層 then
17:   < /div >を出力 (HTML)
18:   < /div >を出力 (HTML)
19: end if
```

クエリ例　例えば，前出の図 4.17 のような sm(768px) をブレイクポイントとするレスポンシブデザインを実装する場合，述べたとおり Bootstrap と Sass のコードはより複雑な記述となったが，SuperSQL では以下のレイアウトの判断が容易な 3 行のクエリにより実装することができる．

```
GENERATE ResponsiveHTML
[image(c.image)@{sm='2/5'}, { c.name ! c.breed }@{sm='3/5'} ]!
FROM cat c;
```

4.6.5　自動生成：レスポンシブレイアウトの自動生成

レスポンシブレイアウトの自動生成は，下記の流れでその処理が行われる．

1. 前処理
2. 配置決定
3. 後処理

それぞれの詳細を以下に示す．まずメインとなる 2 について説明し，その後 1, 3 の説明を行う．

配置決定　レスポンシブレイアウトを自動で生成する試みは, これまでにも例が少なく, 確立した方法は存在していない. この理由として考えられるのは, デバイスに合わせたコンテンツの「配置決定」について考えたときに何をもって「最適」で,「見栄えが良い」のかといった疑問に対する, 明確な基準や, 正解が存在しないからであると考えられる. 幅広い閲覧端末に Web コンテンツを適応させる上で考慮に入れるべき点は, 以下の点などがある. これはほんの一部であり, 全てを網羅しているわけではない.

- 各要素の見た目が極力変わらないこと
  - 大きさ（幅, 高さ）の設定
  - 並び, 配置の設定
  - 余白の設定
- 画面サイズに合ったメニューの構成の設定
  - スマートフォンサイズではメニューを折りたたむ
  - スマートフォンサイズではメニューを簡潔にする
- 要素の表示, 非表示設定
  - あるサイズ以下では, 要素 (サイドバー等) を非表示にする
- 操作方法の統一

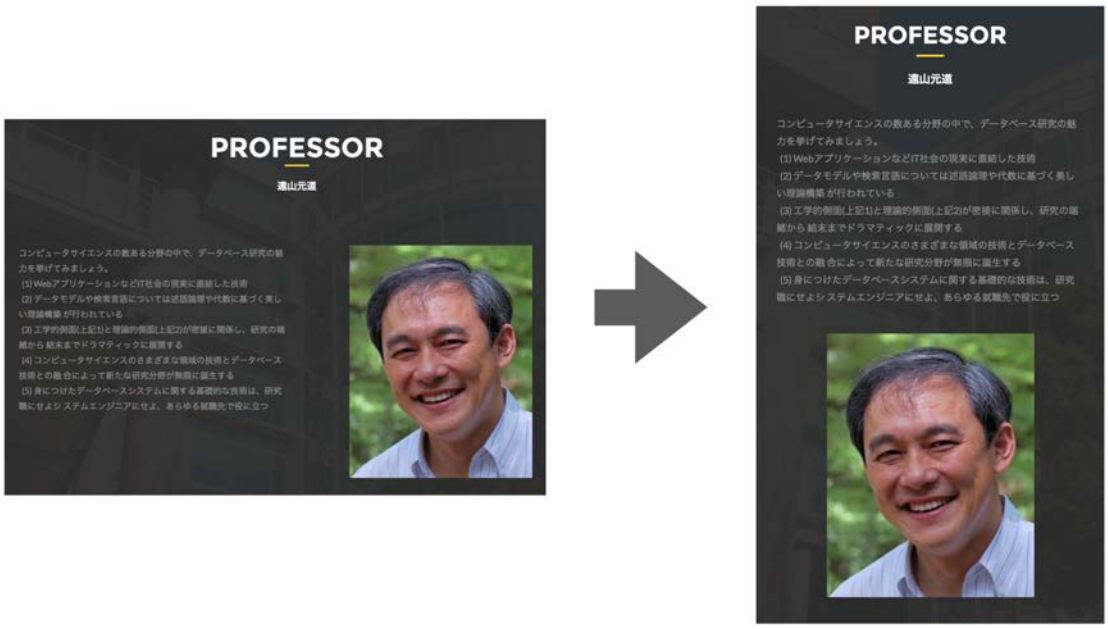


図 4.21: 横並びになっているコンテンツの再配置の例

– タッチデバイスではマウスホバーが出来ない点等を考慮

このように, コンテンツの「配置決定」に関する基準は様々であり, 人が見た場合の「見栄え」を定量的に評価することは非常に難しい問題である.

しかし, 上に挙げた点のうち, 考慮に入れる対象を「コンテンツの大きさや配置の決定」に限定した場合, その配置決定の手段として, Web 上で実際に用いられている代表的な方法は大きく分けて以下の 2 つに絞ることができる.

1. 横並びになっているコンテンツの再配置 (図 4.21)
2. 繰り返し並べられたコンテンツの再配置 (図 4.22)

これら 2 つは, 厳密にはどちらも横並びになっている要素の再配置になるのだが, コンテンツとしての意味的な違いが存在する. 通常の Web 開発では, HTML と CSS の間に, 構造とデザインの分離がなされており, 各要素が横に並んでいるのか, 縦に並んでいるのかといった区別や, 横並びに並んでいる要素が, 同じ性質のものが繰り返し表示されているのかどうかといった意味的な区別をつけることが難しくなっている. しかし, 結合子や反復子を利用することで, 構造を直接記述できる SuperSQL では, 1 の「横並びになっているコンテンツ」は横結合, 2 の「繰り返し並べられたコンテンツ」は横反復という形でクエリ内で記述されているため, この区別を簡単に行えることができる.

また，提案手法のレスポンシブデザイン自動生成の配置決定処理について，下記の方式を考案し検討を行った.

図 4.22: 繰り返し並べられたコンテンツの再配置の例

方式 1 画面サイズに応じて一律で，現在の配置を維持するか縦並びにするかを決定

方式 2 初期のレイアウトを極力維持して配置決定を行う

方式 1 を利用し，スマートフォンサイズの端末画面では「縦並び」配置を使用すると仮定して考えてみる．例えば小さなコンテンツが 3 つ横に並んでいた場合，それらがスマートフォンサイズの端末画面でも横並びのままで十分に閲覧可能であっても本方式では一律で「縦並び」として配置決定されてしまう．方式 2 を用いた場合，小さなコンテンツが 3 つ横に並んでいた場合はそのまま「横並び」を維持し，逆に大きなコンテンツが 3 つ横に並んでいた場合は「縦並び」配置を採用するといった柔軟な対応が可能となる．また，初期のレイアウトを極力維持するため，一律で横並びや縦並びとなるレイアウトが採用されるわけではなく極力，開発者が意図して作成した初期レイアウトに沿った配置が採用される形となる．これらを踏まえ，提案手法ではレスポンシブデザイン自動生成の配置決定方式として方式 2 を採用した．幅広い閲覧環境において, SuperSQL クエリ内で横結合や横反復を用いて記述された各要素の大きさができるだけ変わらないように, それぞれの要素の占有する割合や配置に合わせて自動的に変化されることができれば, それは, デバイスサイズに合わせたコンテンツの配置決定という課題におけるコンテンツの大きさと配置の決定という問題の一つの解決策になると言える.

本手法では, 一番大きいブレイクポイントとして設定されている横幅 1200px 以上の状態でのレイアウトを「基準レイアウト（初期のレイアウト）」とし, 画面サイズを小さくしていった際に各要素の大きさが基準レイアウトの状態のときに最も近い状態を保つよう, 配置決定を行っていく. この際, 配置決定を行う区切りとして, 前出の 1200px

よりも小さい 4 つのブレイクポイントそれぞれにおける配置決定を行う.

横結合の配置決定　横結合の配置決定では, Web ブラウザの画面幅を基準点である画面幅 1200px 以上の状態から 1200px, 992px, 768px, 544px と段階ごとに小さくしていったときに, それぞれの画面幅において, 横結合された各要素の大きさが基準レイアウト (画面幅 1200px) における要素の大きさに最も近くなるように, 各要素のサイズ, 配置を決めていく. 具体的には, 画面幅を小さくした際に, 横結合された要素をそのままの配置で保つ場合と, 一部又は全ての要素を縦に並べた場合を比較していき, 各要素の横幅の変化の差の合計が最も小さいレイアウトを採用する. これを目標に検討した場合，配置決定は下記の関数を用い和の小さい方を採用する形となる.

コスト関数

$|S1-X1|+|S2-X2|+\cdots+|Sn-Xn|$ と $|S1-Y1|+|S2-Y2|+\cdots+|Sn-Yn|$ を比較し，値の小さい方を採用

S: 基準（基準レイアウト）
　Sn: S の左から n 番目の横幅
画面サイズを小さくする際に下記を想定
　X: 比率維持レイアウト (そのままの比率を保つ)
　　Xn: X の左から n 番目の横幅
　Y: 縦並びレイアウト (比率を放棄し縦に並べる)
　　Yn: Y の上から n 番目の横幅

ここからは具体的な例を用いて説明をする. 4 つの要素 A, B, C, D が横結合で並んでいる図 4.23 上部のような基準レイアウトがある場合, 候補レイアウトは, 図 4.23 下部に示される 8 つの配置パターンが考えられる. 提案機構では, 実際に候補レイアウト群に示されるような配置を適用した場合の各要素の幅を取得し, 基準レイアウトの各要素の幅との差が最も小さくなるレイアウトが採用される. つまり, 図 4.23 の例だと, コスト関数は下記のようになり，全ての候補レイアウトにおいて本関数を基に計算を行い和の小さい方のレイアウトを採用する.

$$|(A\text{の幅})-(A'\text{の幅})|+|(B\text{の幅})-(B'\text{の幅})|+|(C\text{の幅})-(C'\text{の幅})|+|(D\text{の幅})-(D'\text{の幅})|$$

また, 提案機構では, 基準レイアウトを作成する上でユーザが指定したコンテンツの配置やサイズ指定を可能な限り保持するレイアウトを作成することを目標としている. そのため, 横結合される要素の順番自体は変えず, 要素にサイズ指定がされている場合は, 配置変えの際に, その比率を保った状態で候補レイアウトを作成していく. 例を用

図 4.23: 横結合の配置決定における候補レイアウトの例

いて説明をする. 横結合された 4 つの要素が順に 1/2, 1/6, 1/6, 1/6 といったサイズ指定がされているとする. このレイアウトから, 横方向に 2 つづつ並ぶような候補レイアウトを作る場合は, 上に並ぶ 2 つの要素はそれぞれ 3/4, 1/4 の比率で, 下に並ぶ 2 つの要素はそれぞれ 1/2, 1/2 の比率で並ぶように配置をする.

横反復の配置決定　横反復の配置決定では, 横結合の配置決定の場合と同様に, 1200px, 992px, 768px, 544px の 4 段階の画面幅それぞれにおいて, 横反復された各要素の大きさが基準レイアウト (画面幅 1200px) における要素の大きさに最も近くなるように, 各要素のサイズ, 配置を決めていく.

GENERATE ResponsiveHTML において, 横反復を用いた場合は, サイズ指定があった場合は指定されたサイズ指定に合わせて, サイズ指定が無かった場合は出力されるデータの数に合わせて, 反復されるコンテンツのサイズが決定されるが, どちらの場合も, 出力結果としては同じサイズの要素が繰り返し表示されるようになる.

よって, 提案機構では, 横結合のときのように配置パターンを元に考えていくのではなく, 各要素のサイズ (占有する割合) を基準として候補レイアウトを作成する. 具体的には, 画面幅を小さくした際に, 反復子によって横方向に反復される各要素の占有する割合をそのままに保つ場合と, 割合を増やして, 各列に並ぶ要素数を減らした場合を比較していき, 各要素の横幅の変化の差の合計が最も小さいレイアウトを採用する.

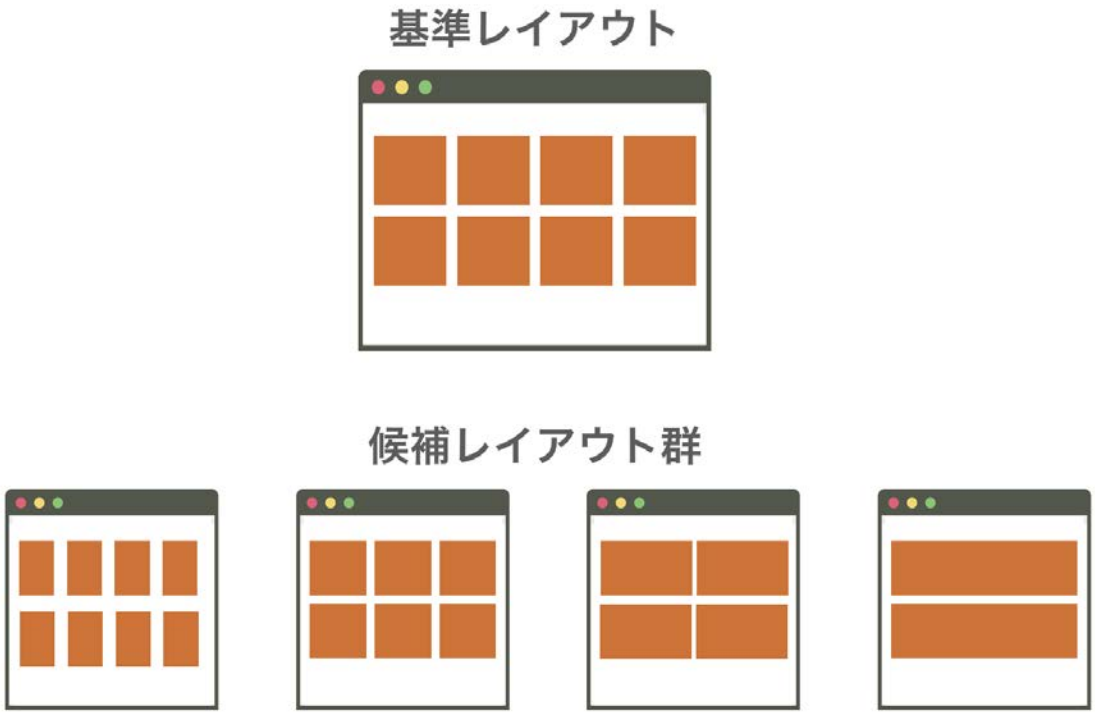


図 4.24: 横反復の配置決定における候補レイアウトの例

例として図 4.24 上部のような基準レイアウトがある場合, 候補レイアウトは, 図 4.24 下部に示される 4 つの配置パターンが考えられる. 提案機構では, 実際に候補レイアウト群に示されるような配置を適用した場合の各要素の幅を取得し, 基準レイアウトの各要素の幅との差が最も小さくなるレイアウトが採用される.

アーキテクチャ　レスポンシブレイアウト自動生成機構は, SuperSQL の追加モジュールとして実装を行った. レスポンシブレイアウト自動生成機構を導入した際の, SuperSQL 本体も含む全体の処理の流れを図 4.25 に示す.

まず最初に SuperSQL のクエリの実行がされると, SuperSQL 側の処理を全て通り, コード生成部 (Code Generator) が基準レイアウトとなるページを生成する. コード生成部は, 基準ページ生成の処理の過程で, 横結合や横反復に関する TFE ID とサイズ指定の情報をまとめており, 基準ページが生成時にレイアウト生成機構 (Layout Fixer) にその情報を渡す. レイアウト生成機構側では, 基準ページをレンダリングし, コード生成部から受け取った横結合と横反復の情報を元に, それぞれの横結合と横反復の配置決定を行うための計算処理を行い, それぞれの TFE ID について, 最適なサイズ指定の情報をまとめていく. 全ての横結合と横反復の配置決定処理が終わると, 最適レイアウトのための TFE ID とサイズ指定の情報がレイアウト生成機構からコード生成部に渡さ

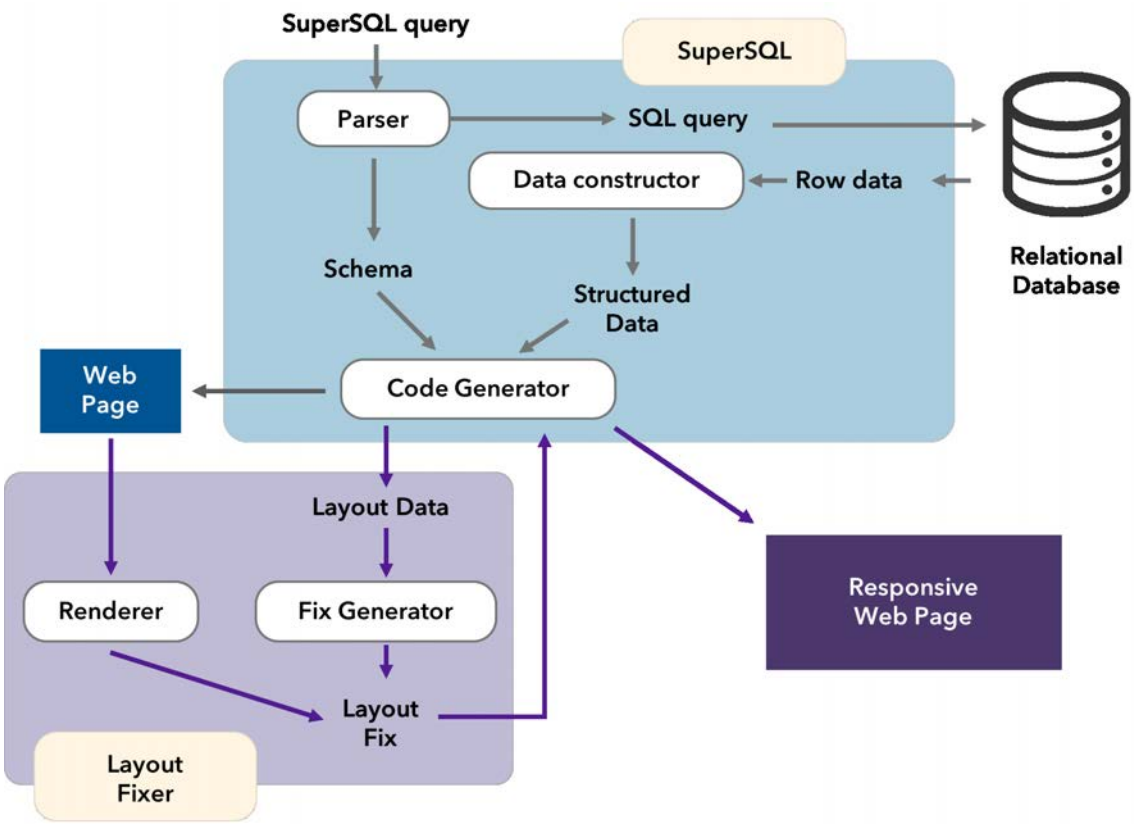


図 4.25: レスポンシブレイアウト生成機構を含んだ SuperSQL 処理系：処理の流れ

れる. SuperSQL では, その情報を利用し, コード生成部のみが再実行されることで, レスポンシブなレイアウトとなったページが生成される.

内部処理　レスポンシブレイアウト生成機構 (Layout Fixer) では, 基準レイアウトの情報を元に, 配置決定のための計算を行っていく. しかし, 基準レイアウトを作成するためのクエリからの情報だけでは, 実際に Web ブラウザでページをレンダリングした際の各要素の大きさを正確に計算することができない. そのため, 提案機構では, Selenium WebDriver[23] を使用して, 基準レイアウトとなる Web ページをレンダリングすることで, 各要素の横幅などの正確な値を取得している. Selenium WebDriver とは, Web アプリケーションの機能テストや結合テストの自動化を実現するブラウザ駆動型のテストツール群である Selenium に含まれるツールの一つで, プログラム上から Web ブラウザを立ち上げ, Web ブラウザを操作することが可能となっている. レスポンシブレイアウトの自動生成機構では, Selenium WebDriver を使って実際に基準ページをレンダリングし, 基準レイアウトの各要素の正確な幅 (px) を取得している. また, それぞれの候補レイアウトを作成する際には, 立ち上げたブラウザ上で, 各要素に適用された CSS の値を変える Javascript のコードを実行している. 作成された候補レイアウトから, 各要素のサイズの取得を行っていくことで, 最適なレイアウトの算出を行っている.

横結合の配置決定アルゴリズムをアルゴリズム 11 に, 横反復の配置決定アルゴリズムをアルゴリズム 12 に示す. どちらも, それぞれの候補レイアウトにおける対象要素の横幅を取得していき, 基準レイアウトとの差が最小となる候補レイアウトを採用していく流れになっている. 最適レイアウトが決定した際には, そのレイアウトを作成するための CSS を Selenium WebDriver で立ち上げたページに適用した上で, 次の対象要素の配置決定に移っていく. このように, それぞれの横結合, 横反復を独立して行うのではなく, 一回ごとの配置決定の結果が次のステップに作用する設計となっている.

アルゴリズム 11 横結合の配置決定

```
for all ブレイクポイント do
  for all 候補の配置パターン do
    for all 横結合の各要素 do
      original_width ⇐ 基準レイアウトにおける要素の横幅
      candidate_width ⇐ 候補レイアウトにおける要素の横幅
      width_difference ⇐ original_width と candidate_width の差
    end for
  end for
  best_arrangement ⇐ width_difference の合計が最小となる配置
  best_arrangement の値を用いて対象の各要素の横幅を変更する CSS をブラウザ上で適用
end for
return 各ブレイクポイントにおける best_arrangement
```

アルゴリズム 12 横反復の配置決定

```
original_division ⇐ 基準レイアウトにおける横反復の分割数
original_width ⇐ 基準レイアウトにおける各要素の横幅
for all ブレイクポイント do
  element_width ⇐ ブレイクポイントの画面幅における各要素の横幅
  for i = 0 to original_division do
    candidate_division ⇐ original_division − i
    {candidate_division ⇐ 候補レイアウトにおける横反復の分割数 }
    candidate_width ⇐ (element_width∗original_division)/(candidate_division)
    {candidate_width ⇐ 候補レイアウトにおける各要素の横幅 }
    width_difference ⇐ original_width と candidate_width の差
  end for
  best_division ⇐ width_difference が最小となる配置
  best_division の値を用いて対象の各要素の横幅を変更する CSS をブラウザ上で適用
end for
return 各ブレイクポイントにおける best_division
```

配置決定の処理順序　提案機構では, レスポンシブレイアウトを自動生成するための方式として, 以下の 2 つの方式を実装した.

- **トップダウン**
  基準レイアウトの HTML の上の階層から順にそれぞれの横結合, 横反復の配置決定を行っていく方式
- **ボトムアップ**
  基準レイアウトの HTML の下の階層から順にそれぞれの横結合, 横反復の配置決定を行っていく方式

これら 2 つの方式の違いは, レイアウト配置決定処理を行う順序だけである. しかし, 4.6.5 項で述べたように, それぞれの階層における配置決定の結果は, 次の階層の配置決定に作用していくため, トップダウン方式とボトムアップ方式では, 自動生成されるレイアウトに違いが生じる. 実際に生まれる違いについては, 第 6 章で行った評価で検証を行っているが, レイアウトの「満足度」は, 出力コンテンツの意味合いやページ作成者の好みにも左右されるため, これら 2 つの方式の優劣を一概に決めるのは難しく, 現状では, どちらの方式でもレイアウト生成を可能としている.

また，トップダウン方式のメリットは，上の階層で加えた変化が下の階層へも作用していくことで全体として無駄のないレイアウトが作られることであり，ボトムアップ方式のメリットは，下の階層で加えた変化が上の階層へも作用していくことで、コンテンツを詰め込んだような窮屈な見栄えになりにくいことである.

配置決定の処理時間に関する検討（前処理）　前出の横結合/横反復における配置決定処理は，一つの横結合で結合される要素数が増えていくと配置の組み合わせが指数関数的に増えていき，このまま単純に総当たりで処理を行う場合は 0-1 ナップサック問題に帰着する．単純に総当たりで処理を行う場合の，横並びに並べるコンテンツの数（結合要素数）毎の配置の組み合わせ数と実行時間を図 4.26, 4.27 に示す.

グラフを見てみると，例えば 7 個の要素の横結合の配置決定には，64 組の候補レイアウトが生成され約 129 秒かかってしまうことになり，これでは実用的とは言えない.

ここで，7 個の要素を横に並べる場合のクエリの記述を見てみる．例えば下記のような TFE を書くと，図 4.28 の表示結果となる.

id,‘名前’, name,‘誕生年’, byear,‘血液型’, btype

また，同じものを，意味のあるまとまりを中括弧で囲った場合の表示結果を図 4.29 に示す．意味のあるまとまりを中括弧で囲った場合，レスポンシブデザインを生成し

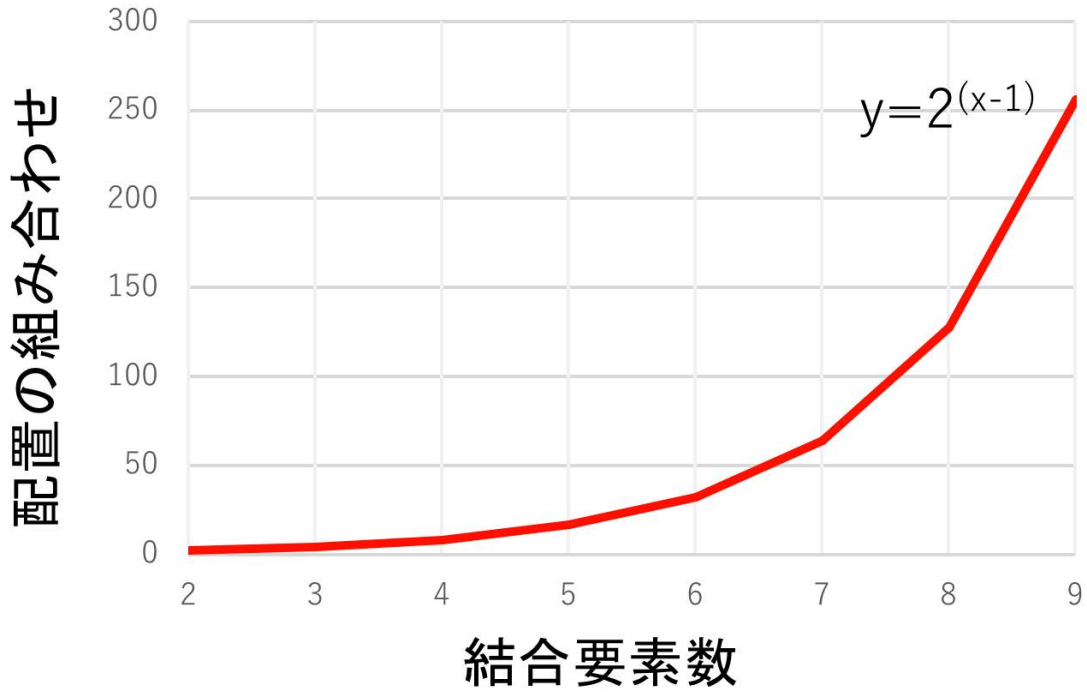


図 4.26: 総当たりに処理する場合（不採用）の横並びに並べるコンテンツの数（結合要素数）と配置の組み合わせ数

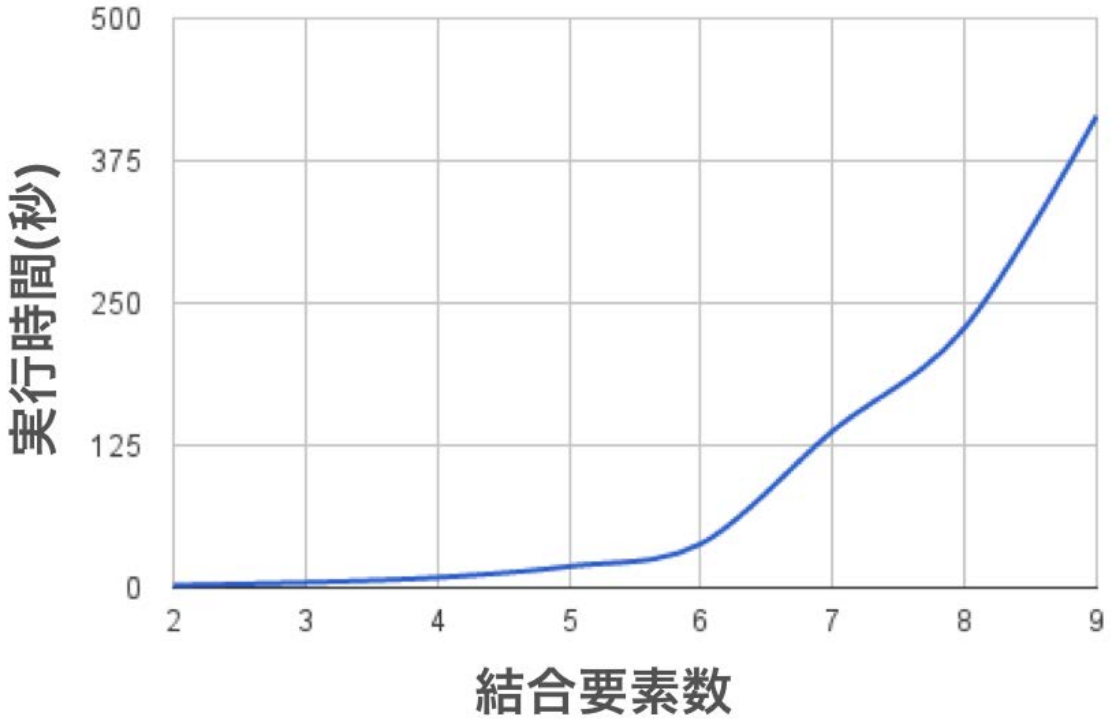


図 4.27: 総当たりに処理する場合（不採用）の横並びに並べるコンテンツの数（結合要素数）と実行時間

1001	名前	田中	誕生年	1950	血液型	A

図 4.28: 属性を単純に並べた場合の表示結果

図 4.29: 属性を中括弧でくくって並べた場合の表示結果

表 4.3: 7 個の要素を横に並べる場合の配置決定の処理時間

	配置の組み合わせ	処理時間 (秒)
中括弧なし	128	129.48
中括弧あり	8	9.88

てスマートフォンで表示させた際，例えば本図のような意味のあるまとまりが同じ行に表示される．

id, {‘名前’, name}, {‘誕生年’, byear}, {‘血液型’, btype}

また，実行時間は中括弧なし/ありで下記の表 4.3 の通りとなる．中括弧ありの場合は中括弧部を一まとまりとして処理するため，組み合わせ数が減り，処理時間が短くなる．

このことから，通常 5 個以上の要素を横結合する場合、開発者は中括弧でまとまりをつくることが推奨される．通常のクエリ作成では，中括弧により意味のある部分を囲い，装飾子でデザインの指定を行う．

しかし，開発者がこちらを怠っていた場合，前出のグラフの通り，処理時間が非常に長くなる可能性がある．そこで，そのようなケースに備えたタイムアウト対策用に，前処理として下記のチャンキングを導入した．

チャンキング：
5 個以上の要素が中括弧で囲われていない状態で横並びに結合されている場合は中括弧を付与し，そののちに前出の配置決定処理を行う．

例えば，3 個/5 個/12 個の要素が中括弧で囲われていない状態で横並びに結合されて

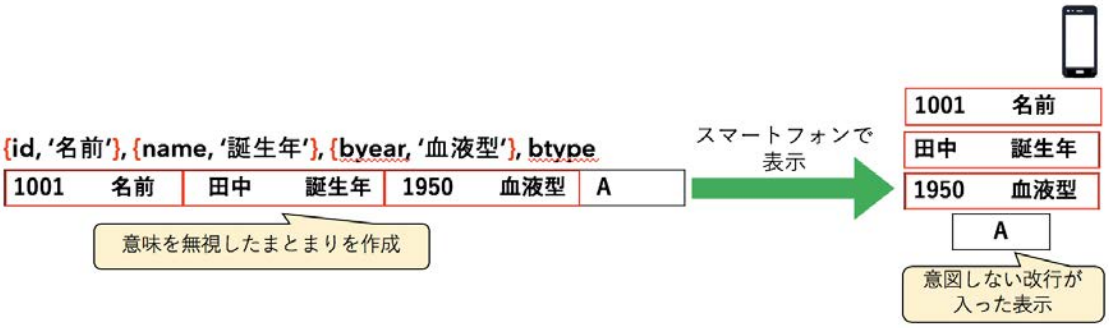


図 4.30: 7 個の横並び要素に前処理「チャンキング」を適用した場合の表示結果

いる場合の例を下記に示す.

A1, A2, A3 → A1, A2, A3
A1, A2, A3, A4, A5 → {A1, A2}, A3, A4, A5
A1, A2, A3, A4, A5, A6, A7, A8, A9, A10, A11, A12
→ {A1, A2, A3}, {A4, A5, A6}, {A7, A8, A9}, {A10, A11, A12}
An: 左から n 番目の要素
→: 前処理

4 個以下の要素が中括弧で囲われていない状態で横並びに結合されている場合は，前処理の前後でクエリは変わらない．5 個の横並び結合の場合は，前処理により左から 2, 1, 1, 1 の 4 つのまとまりとなる（左から 1 つ目と 2 つ目の要素を囲う中括弧が前処理段階で付与される）．12 個の横並び結合の場合は，左から 3, 3, 3, 3 の 4 つのまとまりとなる．まとまりを 4 つとした理由は，3 つだとまとまりとして大きくなりすぎ，また 5 つだと処理速度が 10 秒を超えるため実用的ではないと判断したためである．

また，前処理により中括弧が付与された箇所はラベル付けされ，たとえ中括弧の内側に 5 個以上の要素が入っていても配置決定処理の際にはひとまとまりとして扱う．

前出の 7 個の要素が横並びになった TFE を例に取ってみると，表示は図 4.30 のようになる．

本チャンキングはタイムアウト対策用に実装したものであり，これにより処理時間は実用時間内に収まるが，開発者は，この図のように開発者が意図しない改行が入った表示になる可能性に留意する必要がある．

最後に，一般に Web ページにおける横並びのコンテンツ表示は，縦並びの表示と比べて閲覧者の閲覧時間を 20%以上長くする傾向があり，また誤読に繋がり易くなるた

め [24, 25, 26, 27, 28, 29]，ユーザビリティの観点からも実際の Web ページの開発においては横に並べる要素の数は多くないほうが望ましいと言える．その点から，上記の前処理により対応はしていて使用可能ではあるが，実際の開発においては (もちろん実際に表示させる要素の横幅にもよるが) 例えば 10 個以上の要素を横に並べるのは適切とは言えない．自分も含めた Web ページ開発者に対する注意喚起も兼ねてこの点をここに記す．

また，今回は SuperSQL の特性を利用して上記の処理を採用したが，より厳格に処理速度を考慮する場合は更に処理速度が速くなる他の手法を検討する必要があると考える．それについては今後の課題とする．

配置決定後の処理（後処理） 配置決定が実行され，その後，コンソール（例えば Google Chrome のデベロッパーツールにおける Console 画面）には決定された sm 等の装飾子を含むクエリが提示される．開発者は，そのクエリ情報を基にして現在のクエリを手動で書き換えることも可能である．また，一度配置決定が実行されたクエリは前出のキャッシュ処理によりキャッシュとして保存され，クエリが変更されるまでは再度配置決定は実行されずキャッシュに格納された配置決定後のクエリをベースに 2 回目以降の実行を行う形となる．

第 5 章

埋め込み型SuperSQL:用例

5.1 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. なお, 本節内の図と例で使用している映画の画像と解説は, Yahoo!映画から引用している[1].

本章の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 [30] を使用した. また，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 5.1 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 [30] 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 TFE 指定において，生成するデータの構造やデザインを指定してる.

5.2 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・レスポンシブな映画一覧ページの 4 つの Web ページを例として取り上げる.

[1] 出典：https://movies.yahoo.co.jp/

5.2.1 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_detail.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，5.2.2) 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).

5.2.2 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 5.2 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る. そして，11 行目では記述され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.

5.2.3 装飾子 form の用例:movie_review.html

下記に示す 17 行のコードを 5.2.1 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 5.3 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 insert で囲われた T F E の情報に基づい

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>
```

図 5.1: ジャンル別の映画名一覧ページ (movie_list.html)

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 5.2: レビューを含んだ映画情報ページ (movie_detail.html)

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.

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 ?>
```

図 5.3: 映画レビュー投稿ページ (movie_review.html)

5.2.4 レスポンシブデザイン生成の用例:responsive_movie_list.html

下記に示す 17 行のコードを 5.2.1 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 5.4 のレスポンシブな映画一覧ページを作成している．6〜14 行目の SuperSQL クエリでは，7〜 13 行目の横反復子で囲われた部分が横並びとなる，9 行目で各映画のイメージ画像を表示，10 行目では glink 関数による GET 形式のリンクの作成，11 行目では縦の空白を入れる記述を行っている．12 行目がレスポンシブデザイン生成用の装飾子となり，画面幅が 544px(xs) 以上で 768px(sm) より小さいときは 2 つずつ横並びに表示，画面横幅が 768px(sm) 以上のときは 3 つずつ横並びに表示するよう指定されてる，この場合，画面幅が 544px(xs) を下回った場合は比率を放棄して縦並びとなる．

responsive_movie_list.html の一部

```
1  <h3>注目の <span>映画</span></h3>
2  <?php
3  require_once('./Ssql/ssql.php');
4  ssql_setConfig("./config.ssql");
5  $query1 = "
6      GENERATE ResponsiveHTML
7       [
8         {
9           image(image, './img')@{height=300, width=200, align='center'}!
10          glink(m.title, './movie_detail.html', m.id)!
11          ' '@{height=10}
12        }@{sm='1/2', md='1/3'}
13      ],
14     FROM movie m;
15   ";
16 ssql_exec($query1);
17 ?>
```

図 5.4: レスポンシブな映画一覧ページ (responsive_movie_list.html)

第 6 章

実験・評価

6.1 概要

6.1.1 実験・評価項目

本提案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.

埋め込み型 SuperSQL

- コード量の比較
- 各実行処理における SuperSQL の処理時間の評価
  - 通常の実行（4.5.2 項）
  - データ更新実行（4.5.3 項）
  - 簡易実行 (データ更新なし)（4.5.4 項）
- Web で入手可能なテンプレートに対する埋め込みの可否

レスポンシブレイアウト生成

- レスポンシブレイアウト生成機構の評価
  - コード量の比較
  - 作業時間の比較
- レイアウト自動生成機構の評価
  - 手動生成との比較
  - 処理時間の評価

6.1.2 実験環境

提案手法の評価実験を行った環境は以下のとおりである.

- サーバーサイド
  - VM(実験用)
    - OS: CentOS 7.1
    - CPU: 4 コア

        * メモリ: 32GB
        * DBMS: PostgreSQL 9.4
        * Web 開発フレームワーク: PHP 5.4.16
    - VM が載っているマシン
        * OS: CentOS 7.1
        * CPU: 80 コア
        * メモリ: 1024GB
- クライアントサイド
    - ブラウザ: FireFox 43.0.4

表 6.1: 既存手法と提案手法による実装コード量

	サイト 1	サイト 2	サイト 3	サイト 4	サイト 5
HTML, CSS, JS, PHP	26 行	69 行	97 行	538 行	92 行
埋め込み型 SuperSQL	9 行	22 行	14 行	127 行	25 行
	サイト 6	サイト 7	サイト 8	サイト 9	-
HTML, CSS, JS, PHP	135 行	84 行	226 行	2850 行	-
埋め込み型 SuperSQL	25 行	30 行	42 行	432 行	-

6.2 コード量の比較 (埋め込み型 SuperSQL)

下記の 9 つの Web サイトについて，データベースプレゼンテーション部と入力 (form) 部を，既存手法（PHP，JavaScript，XML，CSS，HTML）を用いて作成した場合と提案手法を用いて作成した場合のコード量の比較を表 6.1 に示す．なお，サイト 5〜9 はデータベースプレゼンテーション部のみを評価対象としている．

サイト 1. ジャンル別の映画名一覧ページ (図 5.1)

サイト 2. レビューを含んだ映画情報ページ (図 5.2)

サイト 3. 映画レビュー投稿ページ (図 5.3)

サイト 4. Community Judge! (2018 年度 情報工学実験最優秀賞) [2]

サイト 5. コンサートグッズ代行購入サイト [3]

サイト 6. チケット出品サイト [4]

サイト 7. パフォーマーとイベントをつなぐマッチングサイト「パフォマ」 [5]

サイト 8. 就職活動支援サイト「Karriere」 [6]

サイト 9. 大規模な OSS「写真共有・管理アプリ Gallery」 (図 6.1) [1]

本評価では，結果としてそれぞれ既存手法と比較して 34%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．特にサイト 3 の form を用いるページでは，コード量は 14%となり大幅なコード量の削減を確認した．また，大規模 Web アプリケーションの一例であるサイト 9（図 6.1, [1]）を提案手法により実装した場合でもコード量は約 6 分の 1 となり，有用であることを確認した．

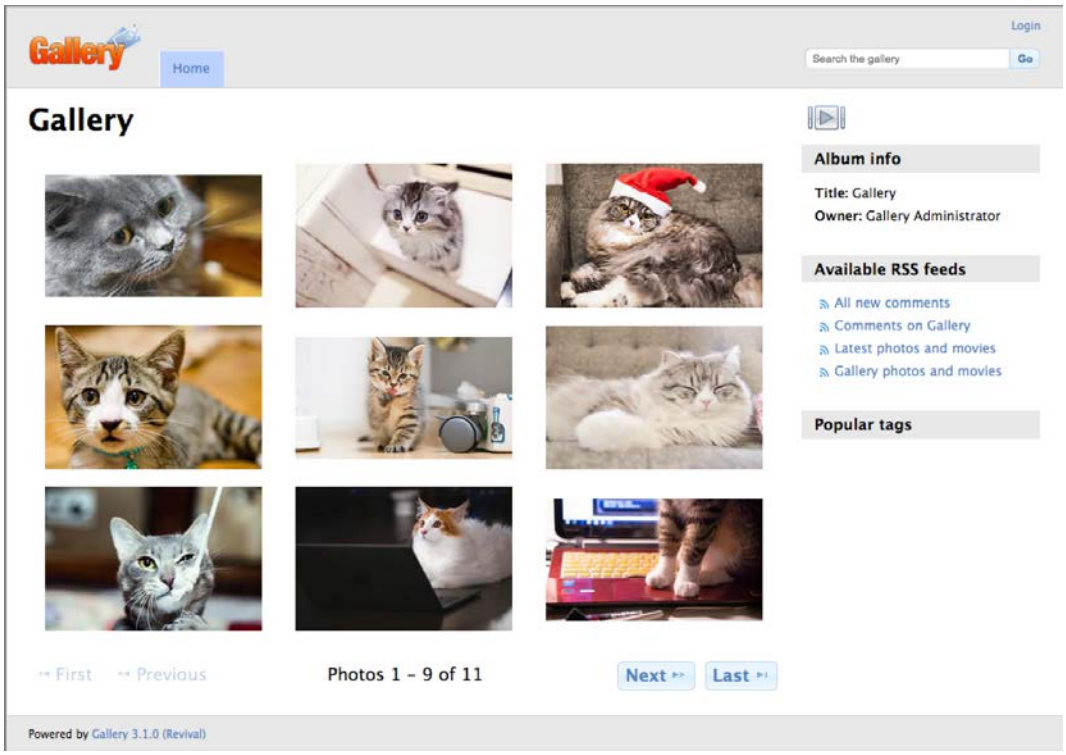


図 6.1: 大規模な OSS「写真共有・管理アプリ Gallery」[1]

6.3 レスポンシブレイアウト生成

6.3.1 レスポンシブレイアウト生成機構の評価

レスポンシブレイアウト生成機構の評価では, コード生成部に新たに実装した, Bootstrap グリッドを用いたレイアウト生成機構の評価を行う. この評価実験では, レイアウトの自動生成機構は用いず, 被験者は装飾指定 (xs, sm, md, lg 装飾子) によって手動でレスポンシブレイアウトを生成している.

コード量の比較 (レスポンシブレイアウト生成) コード量の比較では, 本研究の有用性を示すため, 複数のレスポンシブデザインを用いる Web サイトを実際に作成し, HTML, CSS, Javascript, PHP を用いた従来の Web アプリケーション開発手法とのコード量の比較を行った.

評価に際して作成した Web サイトは以下の 4 つである. これらは全て, 閲覧端末やブラウザの画面幅に合わせて動的に表示を変更するレスポンシブなレイアウトとなっている.

サイト 1. 企業のホームページを模した 1 ページの Web サイト (図 6.2)

表 6.2: 実験評価用 Web サイトの機能一覧

	1	2	3	4
データの表示	○	○	○	○
レスポンシブ・デザインへの対応	○	○	○	○
複数ページの構造		○	○	○
コンテンツのページ分割			○	○
フォームを使ったデータの入力			○	○
動的なデータ表示			○	○
ログイン機能				○

サイト 2. 映画の情報を閲覧できる復数ページの Web サイト (図 6.3)

サイト 3. The Movie DB[1]を再現した Web アプリケーション (図 6.4)

サイト 4. Newsweek[2]を再現した Web アプリケーション (図 6.4)

それぞれの Web サイトに実装された機能の一覧表を表 6.2 に示す. サイト 1, サイト 2 は閲覧専用の Web ページである. サイト 3, サイト 4 には Web アプリとしての様々な機能を持たせている.

[1]https://www.themoviedb.org/
[2]http://europe.newsweek.com/

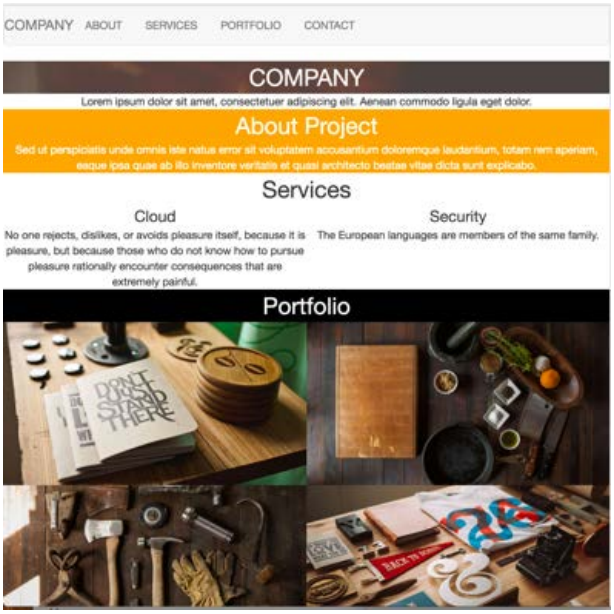


図 6.2: Web サイト 1

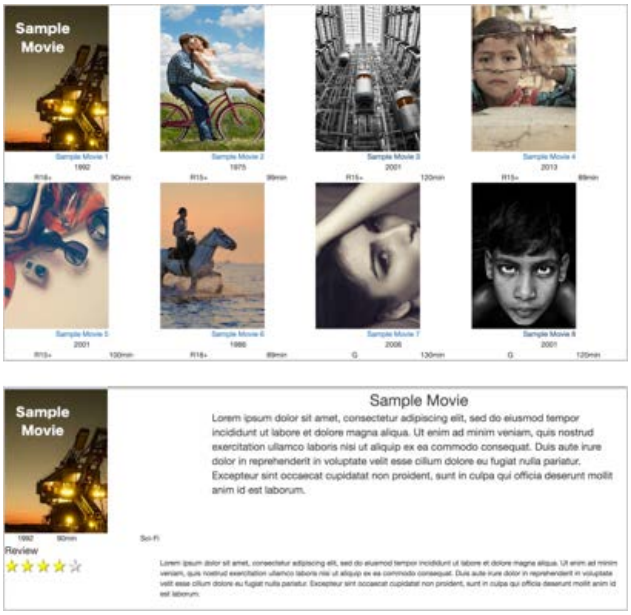


図 6.3: Web サイト 2

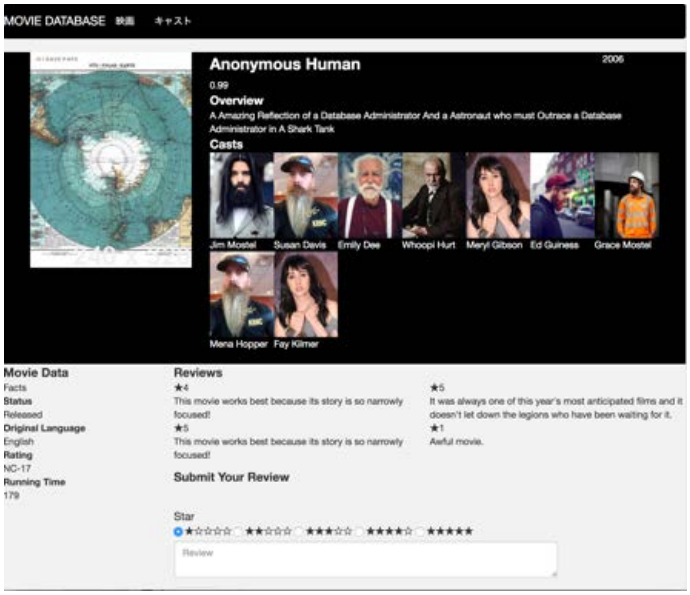


図 6.4: Web サイト 3

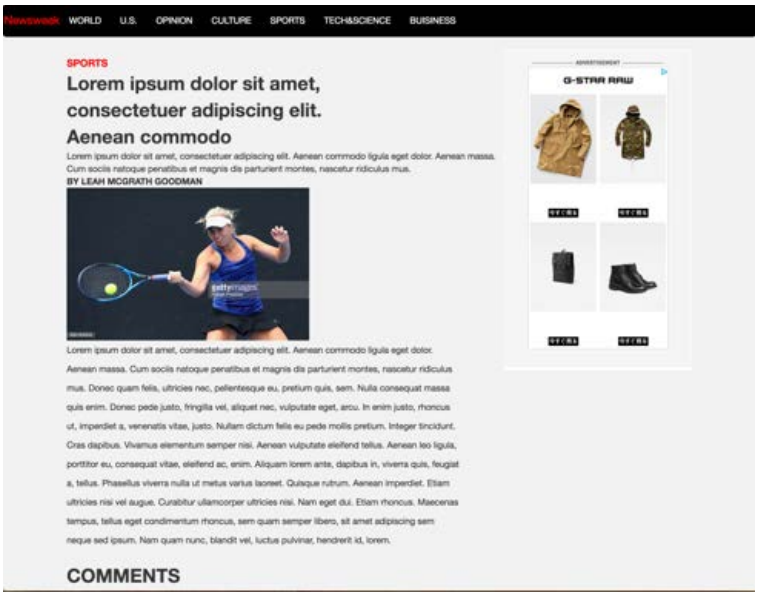


図 6.5: Web サイト 4

表 6.3: 既存手法と提案手法による実装コード量

	サイト 1	サイト 2	サイト 3	サイト 4
HTML, CSS, JS, PHP	137 行	139 行	955 行	930 行
SuperSQL	30 行	37 行	136 行	85 行

コード量の比較結果を表 6.3 に示す. どの Web サイトにおいても, SuperSQL を用いて Web サイトを作成した場合の方がコード量が大幅に削減できている. 具体的には, サイト 1, サイト 2 の Web ページ生成では PHP と比べて 21~27%のコード量となっており, サイト 3, サイト 4 の Web アプリケーション生成では PHP と比べて 10~14%のコード量となった. このように, Web サイトの構造が複雑化し, アプリケーションとしての機能が増えれば増えるほど, コード量の差が顕著になる. これは, 数十行の Javascript を記述する必要があるコンテンツのページ分割や, 数十行の PHP を記述する必要があるフォームの作成及び処理の機能などを, SuperSQL では数行（1〜10 行程度）で記述可能なため, こうした機能が多いアプリの作成ほど, コード量に差が出やすくなるためだと考えられる.

作業時間の比較　作業時間の比較では, 6.3.1 項で作成したサイト 1 とサイト 2 を, 被験者に再現してもらい, HTML, CSS, Javascript, PHP を用いた従来の Web アプリケーション開発手法と SuperSQL を使った手法での作業時間の比較を行った. 被験者は, これまでに 1 回以上, PHP 及び SuperSQL を用いて Web アプリケーション作成を経験したことのある情報工学科の修士課程の学生 6 人である. 評価実験の流れを以下に示す.

1. サイト 1, サイト 2 を作るために必要な以下の知識を被験者に学んでもらう
   - Bootstrap のグリッドの仕組み
   - Bootstrap を導入した場合の HTML の書き方
   - xs, sm, md, lg 装飾子を使用した SuperSQL でのレイアウトの作成方法
   - navbar などの UI コンポーネントを生成するための関数
2. 6 人の被験者の内, 3 人に SuperSQL を用い Web サイト 1 を作成してもらい, 別の 3 人には PHP を用いて Web サイト 1 を作成してもらう
3. SuperSQL で Web サイト 1 を作った被験者には PHP で, PHP を最初に使った被験者には SuperSQL で Web サイト 2 を作成してもらう

それぞれのサンプルサイトにおいて, 作成のためにかかった平均の作業時間を表 6.4 に示す. SuperSQL のレスポンシブ Web アプリケーション生成機構を用いた場合, コード量だけでなく, 作業時間も大幅に削減できていることが分かる. 実験後に答えてもらったアンケート調査では, SuperSQL の利点として以下の点が上がった.

表 6.4: 作業時間の比較結果

	サイト 1	サイト 2
HTML, CSS, JS, PHP	60 分（74 分, 52 分）	85 分（105 分, 69 分）
SuperSQL	25 分（28 分, 19 分）	35 分（43 分, 32 分）

（最大作業時間, 最小作業時間）

- データベースとの接続, データの取得などが非常に簡単
- 反復の処理が反復子で済むのが楽だった
- 通常の Bootstrap と比べて, 分数で各要素の大きさを指定できるのが直感的だった
- row と column の対応関係を自動で処理してくれるので, レイアウトの組み立てが楽だった
- sm などの装飾子で各要素の割合を分数で指定したが, 全てに記述しなくても, 残りのグリッドを分配してくれるので, 便利だと感じた
- navbar が div タグの構造を特に意識することなく, 目的の形で生成できた

また, SuperSQL で不便だった点や PHP の方が便利だと感じた点としては以下のような意見があった.

- 中括弧と結合子の関係を上手く記述できないと, 簡単にレイアウトが崩れてしまうこと
- 細かい操作, デザイン指定ができない又はしづらい

また, PHP か SuperSQL のどちらか片方しか使えないとするとどちらを利用したいか, という質問には, 6 人中 4 人が PHP を選択した. この理由として, 細かいデザインや操作の変更が SuperSQL では不可能であることから, 作り込んでいくことを考えると PHP を選ぶといった意見が多かった. しかし, 両方とも使えるのであれば, 全体のレイアウトや基本的な機能を SuperSQL で作成し, その後の細かい修正を生成された PHP や HTML に加えていく方法が最も便利な使い方であるという意見もあった.

6.3.2 自動レイアウト生成機構の評価

レスポンシブレイアウトの自動生成機構の有用性を評価するために, 実際に Web サイトのレイアウトを作成し, それを手動でレスポンシブ対応した場合とレスポンシブレイアウトを自動生成した場合の比較を行う実験を行った. 具体的には, 被験者（情報工学科の修士課程の学生）3 人に, 合計で 38 個のレスポンシブレイアウトの装飾子指定を SuperSQL クエリで記述もらい, 以下の 2 つのレイアウトの比較を行った.

- 自動生成：レイアウトの装飾子指定に xs 装飾子のみを用いて生成した基準レイアウトを自動的にレスポンシブレイアウトに変換したもの
- 指定生成 : レイアウトの装飾子指定に xs,sm,md,lg 全ての装飾子を利用して, 被験者が手作業でレスポンシブレイアウトを作成したもの

比較方法としては, クエリ内の 1 つ 1 つの横結合, 横反復の装飾指定について, 以下の点を評価した.

- **一致度**：装飾指定が一致している割合
- **満足度**：自動生成レイアウトの挙動に対する製作者の満足の度合い
  - **満足**：自動生成されるレイアウトの方が好ましい
  - **不満**：自動生成されたものでは, 手動生成に比べて見栄えが悪い

また, レスポンシブレイアウトの自動生成の方法に関して, 以下の 2 つの方式を使用し, その比較も行った.

- **トップダウン**
  基準レイアウトの HTML の上の階層から順にそれぞれの横結合, 横反復の最適化を行っていく方式
- **ボトムアップ**
  基準レイアウトの HTML の下の階層から順にそれぞれの横結合, 横反復の最適化を行っていく方式

結果を表 6.5 に示す. 全体としては, 平均の一致度が約 74.8%, 満足度「満足」を得た割合が約 85.8%, という結果になり, 自動生成されるレイアウトが多くの場合で満足度の高いレイアウトを生成できていることが分かった.

横結合に関しては, トップダウン方式よりも, ボトムアップ方式の方が一致度, 満足度, 共に高い結果となった. これは, トップダウン方式では, 上の階層で加えた変化が下の階層へも作用していくことで, 全体として, 無駄のないレイアウトが作られる反面, コンテンツを詰め込んだような窮屈な見栄えになることもあり, それが不満につながる場合があったからだと考えられる.

横反復については, 一致度は横結合の場合よりも低いが, 満足度は横結合の場合よりも高いという結果となった. 画面幅の大きい領域では, 横反復されるそれぞれの要素のサイズの変化が見た目には現れにくいことが多く, レスポンシブレイアウトを手作業で作っていく際に, 細かいサイズ指定を行わないで済ます場合が多い. それに対し, 自動レイアウト生成機構では, こうした手作業で見逃してしまうような細かい最適化をしっかり行っていくことから, 自動生成と手動生成のレイアウトが「一致」はしないが, むしろ自動生成されたレイアウトの方が好ましい, ということが起こるといった結果となった. また, トップダウン方式とボトムアップ方式の比較では, 上の階層での変化を

表 6.5: レスポンシブレイアウト自動生成機構の評価実験結果

		一致度	満足度	
			不満	満足
横結合	トップダウン	72.0%	25.4%	74.6%
	ボトムアップ	85.2%	9.1%	90.9%
横反復	トップダウン	73.6%	0%	100.0%
	ボトムアップ	68.4%	21.0%	79.0%

考慮に入れないボトムアップ方式では, 画面を小さくした際の要素の幅の割り振りが大きすぎる場合があり, トップダウンよりも一致度, 満足度共に低い結果となった.

6.4 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できるかを以下に示す項目ごとに評価する.

- 使用するデータベースのタプル数別
- クエリ内の属性数別
- クエリ内のグルーピング数別

また，結果グラフ中の①〜③はそれぞれ下記に対応している.

① 通常の実行（4.5.2 項）

② データ更新実行（4.5.3 項）

③ 簡易実行 (データ更新なし)（4.5.4 項）

6.4.1 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 100 タプルから 100 タプルずつ 1000 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 6.6 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通常の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わかる. 平均すると約 20msec の改善が見られた. 次に, キャッシュがあり, かつ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 への問い合わ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わかる. タプル数 100 では, 大きな削減は見られないが, 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わかる.

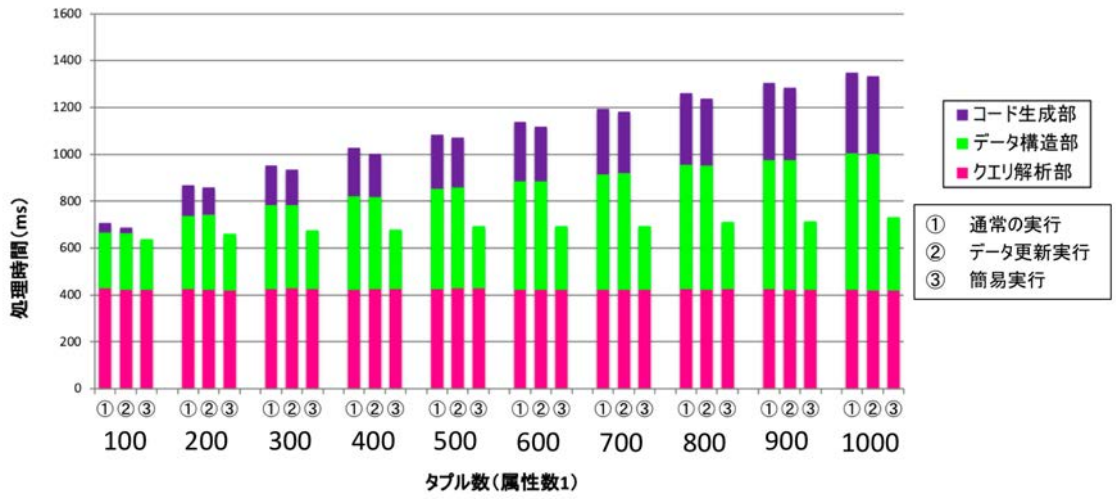


図 6.6: タプル数別処理時間の比較

6.4.2 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 から 8 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 6.7 に示す. 6.4.1 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 20ms 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.

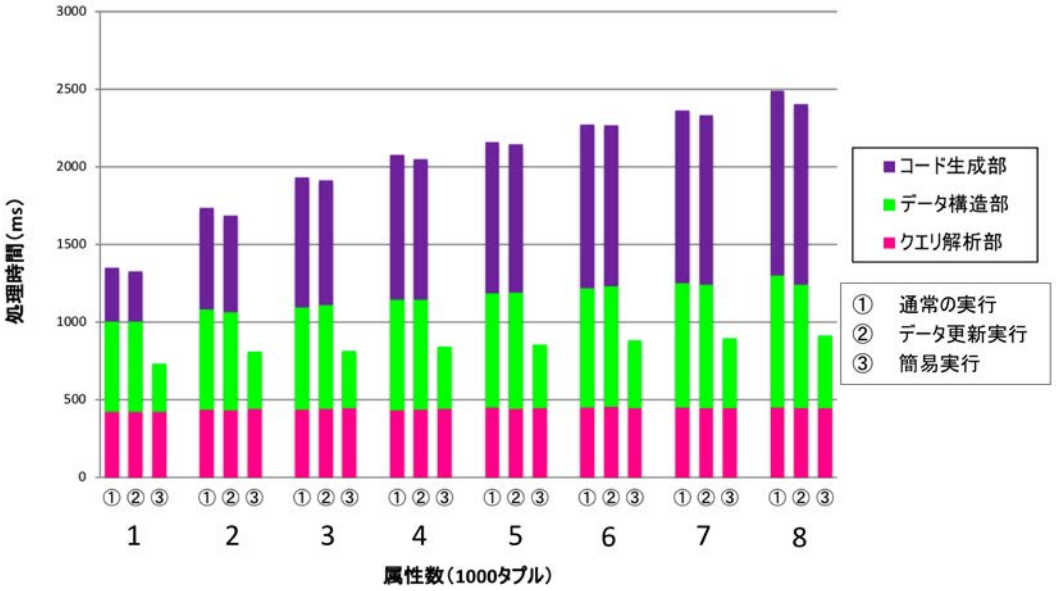


図 6.7: 属性数別処理時間の比較

6.4.3 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 1 から 3 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 6.8 に示す. 6.4.1 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 20ms 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わかる. 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.

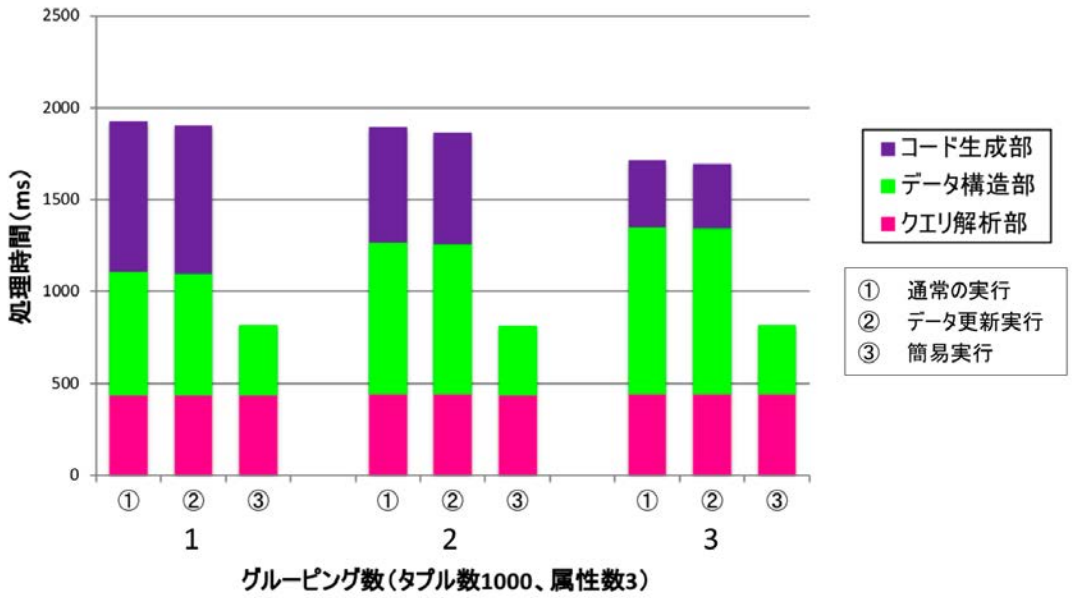


図 6.8: グルーピング数別処理時間の比較

6.4.4 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 20ms の処理時間の削減を行うことができた. 現在の 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 ファイルに比べ, 非常に小さいため, 処理時間の削減率は小さくなってしまっているが, 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.

6.5 Webテンプレートに対するSuperSQLの埋め込み可否

6.5.1 評価方法

提案手法の有用性を評価するために, Web から入手することができる様々な HTML テンプレートに対して, 提案手法を用いて, 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, 以下に示すフレームワークを使用したテンプレートを 90 件, フレームワークを使用していないシンプルテンプレートを 55 件, 計 146 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（91 件）
  - Bootstrap（15 件）
  - Material Design Lite（5 件）
  - Pure（8 件）
  - Foundation（24 件）
  - SkyBlue CSS（5 件）
  - HTML5 Boilerplate（3 件）
  - UIkit（6 件）
  - その他フレームワーク（25 件）

6.5.2 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 6.6 に示す. 表 6.6 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.

結果として，ほとんどの Web テンプレートへ埋め込み可能であることを確認した. 埋め込み表示がうまくいかなかった 1 件の Bootstrap（レスポンシブデザイン）のテンプレートは，テンプレートのコードの最下部に古い Bootstrap のライブラリ読み込み処理が記述されているものであった．確認したところ，提案手法の埋め込みコードよりも下方に 2015 年以前の Bootstrap ライブラリの読み込み処理が記述されている場合は，埋め込むとページデザインが崩れる可能性があることが分かった．2016 年以降の

表 6.6: Web テンプレートに対する埋め込み可否

使用したテンプレート	埋め込み
Bootstrap（レスポンシブデザイン）（15 件）	93.3%（14 件）
Material Design Lite（5 件）	100%
Pure（8 件）	100%
Foundation（24 件）	100%
SkyBlue CSS（5 件）	100%
HTML5 Boileropate（3 件）	100%
UIkit（6 件）	100%
その他フレームワーク（25 件）	100%
シンプルテンプレート（55 件）	100%
計 146 件	99%（145 件）

Bootstrap ライブラリを使用している場合は，問題なく本提案手法による埋め込みとの協調が可能である．

第 7 章

関連技術・関連研究

7.1 関連技術

7.1.1 埋め込み型 SuperSQL の関連技術

近年，動的 Web サイトの開発でよく用いられているのが，サーバサイドプログラミング言語である PHP[31] や Java[32]，Ruby on Rails[33]，ASP.NET[34] である．特にサーバサイドプログラミング言語の中での使用率が約 82%[7] である PHP は，全世界で広く使用されており，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．しかしながら，これらの言語を用いて動的 Web サイトの開発を行う場合，HTML 等の他の言語が必要不可欠になってしまうほか，多量のプログラムや Web コンテンツのレイアウト変更に伴う大幅コードの変更が不可欠となってしまう．また，開発者による HTML コード等の記述ミスによって生成される無効な HTML が引き起こすエラーや，ブラウザによるこのようなエラーの修正の影響でレスポンスが低下したり，脆弱性が高まってしまうといったこともあげられる．

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[35] のライブラリである jQuery[36] を用いて解析し，あたかもデータベースを使用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 CSS の分離の他に，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を始めとし，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だけで実現できるといった利点があげられる．

7.1.2 レスポンシブレイアウト生成の関連技術

Web アプリケーション開発　Web アプリケーションは，主に PHP[31]，Ruby[37]，Perl[38] などのスクリプト言語を用いて開発されることが多い．Web アプリケーションは，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などのウェブサイトを作成するために広く使われており，その開発者も専門家のみではなく一般に広がっている．しかし，Web アプリケーション開発には HTML，CSS，および JavaScript を使用したフロントエンド側の開発の知識から，プログラミング言語，データベース操作などのサーバサイドの知識など，幅広い知識が必要となっており，簡単とはいえない．こうした Web アプリケーション開発を支援するために，近年では多くの Web アプリケーションフレームワークが登場している．

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 アプリケーションや Web サービスの開発をサポートするために設計されたアプリケーションフレームワークである．Web アプリケーション開発の労力を削減することが目

的だが，その機能はフレームワークによって様々である．多くのフレームワークで採用されているものはデータベースアクセスのためのライブラリやテンプレートといったものである．

現在，様々なプログラミング言語のために多くの種類のウェブアプリケーションフレームワークが存在する．例えば，PHP の場合，CakePHP[39]，Lavarel[40]，Biscuit[41]，Symfony[42] が広く使われている．Java 用には，Play Framework[43]，Tapestry [44]，Velocity[45] などが，Perl 用には，Ark[46]，Catalyst[47]，Mojolicious[48] などが，Python 用には Django[49]，TurboGears[50]，Flask[51] などが存在する．ほぼ全てのフレームワークに関して，データベースへのアクセスはオブジェクト関係マッピングを用いている．オブジェクト関係マッピングでは，オブジェクト指向言語で扱う「オブジェクト」と「リレーショナルデータベース（RDB）のレコード」をマッピング（対応付け）することで，プログラミングでのデータベース操作にかかわる煩雑な作業を軽減し，拡張性・柔軟性を持ったアプリケーションの構築をサポートしている．オブジェクト関係マッピングでは，データベースにマッピングされたオブジェクトの集合の内容を含む HTML ファイルを出力する HTML ファイルを出力するためのロジックを手続き的に記述する必要があるが，SuperSQL では，クエリ内で非手続き的に表現されるレイアウト構造に従って HTML ファイルが自動的に生成される．

レスポンシブデザイン　レスポンシブデザインは，デスクトップパソコンやタブレット，スマートフォンといった幅広いデバイスのいずれに対しても，外観や操作方法が最適化されたサイトを制作するためのウェブデザインの手法である．Google の定義では,「ユーザーのデバイスに関係なく，同じ URL で同じ HTML コードを配信しますが，画面サイズに応じて（つまり「レスポンシブ」に）レンダリングを変えることができるデザイン」をレスポンシブデザインと呼ぶ．

レスポンシブデザインの実現には，

- ページ要素にピクセルやポイント等の絶対単位ではなく，百分率等の相対単位を使用するフルードグリッド (fluid grid)
- コンテナ要素の外に画像がはみ出て表示されるのを防ぐため，相対単位を使用するフルードイメージ (fluid image)
- 横幅・高さ・色などのメディア特性を用いてスタイルシートの適用範囲を制御するメディアクエリ (media queries)

を利用する．

レスポンシブデザインは，想定されるデバイスごとに Web サイトを作成していく手法に比べて，端末や OS に依存しないため，新しく登場する閲覧環境にも対応できる点や，HTML ファイルが 1 つになることで，コンテンツの更新が容易である点，URL が

統一されるため，検索エンジン最適化 (SEO) において優位である点などの利点がある．しかし，画面サイズごとに細かいスタイルの定義が必要となるレスポンシブデザインを一から作ろうと思うと，HTML と CSS の深い知識と入念な微調整が必要となり開発のコストが高くなってしまうという問題点も存在する．

CSS フレームワーク　前述した CakePHP などの Web アプリケーションフレームワークの機能は，サーバサイドなどのバックエンドの開発の簡略化に焦点をあてているものが多く，フロントエンド側の動きやデザインなどは，ユーザに委ねられている場合がほとんどである．そこで，最近では，Web デザインのプロセスを簡素化するためのフロントエンド用のフレームワークも多く開発されており，幅広く普及している．これらのフレームワークは，CSS や Javascript のパッケージとして提供されており，「CSS フレームワーク」と総称されている．体系化されたレイアウト構成の仕組みを基盤としたコーディングルールに従うことで，Web ページのレイアウト構造を容易に組み立てることができるようになっている．

現在，多種多様なフロントエンド開発用のフレームワークが存在しているが，中でも最も人気を集めており普及しているのが Bootstrap[17] である．Bootstrap は，現在 Web 全体の約 14%に使用されており，そのシェアは増え続けている [52, 53]．他には ZURB 社が開発している Foundation [54] や，Google の Material Design Lite [55] が注目されている．これらのフレームワークは，レスポンシブデザインを構築するための支援機能が備わっている他，ナビゲーションバーやボタンなど，Web アプリを構成する各種 UI コンポーネントを統一されたデザインのテンプレートとして提供している．

CSS フレームワークは，CSS の深い知識を必要とせずとも，Web ページのレイアウト構造を構成できる利点があるが，それぞれのフレームワークのテンプレートや初期設定をカスタマイズをする際に，非常に手間と知識が必要となり，柔軟性が低いといった問題もある．

CSS プリプロセッサ　近年，レスポンシブデザインの普及や，Web サイトの大規模化に伴い，CSS のコーディングが複雑になってきている．しかし，CSS は，コードの衝突が起こりやすく，拡張性や保守性に欠けているといった問題などがあり，そうした問題に対処するために，CSS の拡張言語である CSS プリプロセッサが開発されている．その中でも，開発が盛んで，人気を集めているものが Less と Sass である．Less や Sass では，セレクタの入れ子や，セレクタの継承，変数，条件分岐や繰り返し処理，mixin などを可能にする機能が実現されており，CSS コーディングの効率化や，メンテナンス性の向上を図っている．

コンテンツ管理システム　Web ページ制作を行う方法として，コンテンツ管理システム（CMS）[56, 57] を用いた方法も存在する．CMS を用いた Web ページ制作は，HTML や CSS ファイルに触れる必要がほとんどないため，専門的な知識がない初心者でも Web サイトが作れる方法として人気を集めている．具体的な例としては，WordPress[58]，Joomla[59]，Drupal[60] などが幅広く使われている．これらのシステムでは，開発者はそれぞれの CMS に用意された，初心者でも分かりやすい設定画面でウェブサイトのコンテンツの作成と管理を行い，デザイン面においては，用意されているデザインテンプレートを適用する場合がほとんどである．しかし，これらのデザインテンプレートの多くが前述した CSS フレームワークを利用していることから，CSS フレームワークと同様に，デザイン面でのカスタマイズにおけるコストが高く，柔軟性が低いといった問題がある．

7.2　関連研究

7.2.1　埋め込み型 SuperSQL の関連研究

Rode らが提案している Click[61] 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ーを対象としている．

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 [62]．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．

7.1 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 [63]．また，Ducasse らが提案している Seaside[64] 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出来る．

SuperSQL(独立型 SuperSQL) の関連研究としては， SuperSQL の言語機能の拡張 [65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 12]，SuperSQL 処理系の改良 [76, 77, 78, 79] がある．独立型 SuperSQL では，クエリ単体でメディアの生成を行うことができ，例えば 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 [80] では，本稿で述べた埋め込み型 SuperSQL の導入部について提案している．

7.2.2 レスポンシブレイアウト生成の関連研究

多様なデバイスへの Web ページの最適化に関する研究は，あらゆる形で提案されてきた．Koehl らの提案する m.Site[81] は，既存のデスクトップ PC 向けの Web サイトのコンテンツを，より小さな画面のモバイル端末に適応させる方法をとっている．m.Site では，コンテンツを再構成するビジュアルツールと，再設計されたページを動的に生成するプロキシサーバーを提供している．

Sinha と Karim は，開発者にとって負担となるレスポンシブデザインの設計を支援するために，モックアップベースの設計ツール [82] を提案している．このツールでは，WYSIWYG エディタで描画されたモックアップを，可変レイアウトへと変換する仕組みを実現することで，レスポンシブデザインの設計を直感的に行えるようにしている．

また，Sinha は，このモックアップツールの他に，レスポンシブデザイン作成支援のための推薦ツールである DECOR[83] を提案している．このツールは，基準となるデザインと，最低限満たすべきユーザ指定の制約を元に，各デバイスサイズ向けのデザインの候補をランク付けしてユーザに提案する．このシステムでは，ユーザーが提供する制約と要件を満たすデザインは，それぞれのデバイスサイズに「適合」できているという考えに基いて，レイアウトデザインの候補を作っていく．

7.3 データベース出版/データベースプレゼンテーション

データベースに格納されているデータから Web（HTML），XML，PDF などの媒体ファイルを生成することをデータベース出版/データベースプレゼンテーション（Database publishing / Database presentation）と呼ぶ．

データベース出版/データベースプレゼンテーションを行うアプリケーションは，Adobe FrameMaker[84]，Adobe InDesign[85]，Datalogics Pager[86]，QuarkXPress[87]，Xyvision[88]，Arbortext Advanced Print Publisher[89]，priint:suite[90] などがあげられる．これらは DTP(Desktop publishing, [91]) ソフトウェアと呼ばれており，一般に，PC 上で雑誌やパンフレットなどの「紙媒体」を生成対象とした原稿作成，編集，デザイン，レイアウトなどを行うことが可能である．

データベース出版（Database publishing）という言葉は 1981 年に誕生し [92, 93],「紙媒体」や「マルチメディア (Video disc, CD-ROM など)」を生成対象とした出版/プレゼンテーションに用いるアプリケーションの研究が始まった [94, 95, 96, 97]．1990 年代以降は，既存言語のコードをベースとしたクエリ等によるデータベース出版の研究

も行われるようになり，TeX コードをベースとした「紙媒体」のデータベース出版システム [98]，OLAP の概念をベースとしたクエリによる，Web ブラウザ上で対話的に情報を検索するシステム [99] , XML ベースのクエリを SQL リクエストに変換し，表形式のクエリ結果を受け取って構造化し XML ドキュメントを返すシステム [100, 101] などが登場した．

近年では，データベース出版におけるプライバシーリークを検出するアプローチ，データのプライバシーを保護する手法など，特にプライバシーに関する研究 [102, 103, 104, 105, 106] が盛んに行われている．

7.4 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 7.1 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，7.4.1 項から 7.4.4 項に示す．

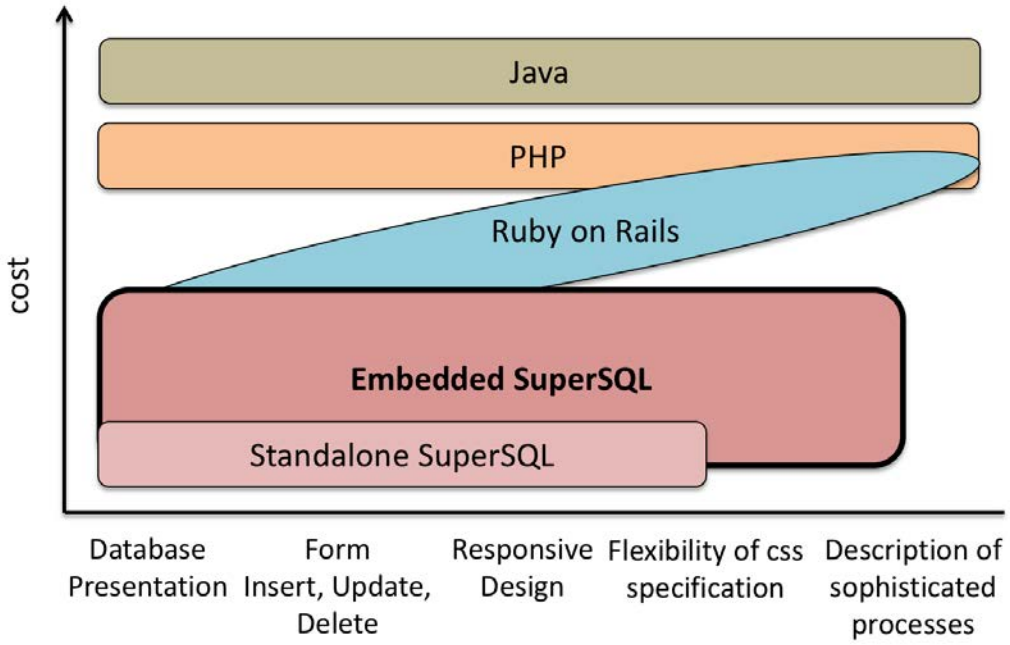


図 7.1: 各言語を用いた場合の記述可能な処理とそのコスト

7.4.1 関連技術・関連研究との比較

手続き型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 [61]，テーブルレイアウトを最適化する手法 [62]，プログラムのエラーを修正する手法 [63] 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[64] 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. 既存のデスクトップ PC 向けの Web サイトのコンテンツをより小さな画面のモバイル端末に適応させる m.Site[81] は，モバイルデバイス向けに Web サイトを再設計するコストを削減しているが，複数のデバイスサイズにワンソースで対応できるレスポンシブなページを作成することを目的としていない．開発者にとって負担となるレスポンシブデザインの設計を支援するためのモックアップベースの設計ツール [82] は, 初心者にも分かりやすい方法で，可変レイアウト（フルードグリッド, フルードイメージ）の作成を実現している反面，CSS メディアクエリを用いて，画面幅に合わせたスタイルの分岐は実現できておらず，生成されるレイアウトが完全に「レスポンシブ」なデザインであるとは言い難い．レスポンシブデザイン作成支援のための推薦ツールである DECOR[83] では，ユーザーが提供する制約と要件を満たすデザインは，それぞれのデバイスサイズに「適合」できているという考えに基いて，レイアウトデザインの候補を作っていく．提案手法では，基準となるデザインのみを必要としており，ユーザにデザインの制約を記述する負担が発生しない．

7.4.2 既存 Web テンプレートを用いた開発手法との比較

提案手法では，TFE 内で装飾子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 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 (2.2.3 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.

7.4.3 既存データベース出版/データベースプレゼンテーションとの比較

既存のデータベース出版/データベースプレゼンテーションの多くは,「紙媒体」を生成対象としている．OLAP の概念をベースとしたクエリによる Web ブラウザ上で対話的に情報を検索するシステム [99] は，生成対象を「HTML」としておりその点では提案手法と同じであるが，その機能は情報の検索に限定されており，また提案手法のように任意のレイアウトによる出力が行えるわけではない.

また，既存データベース出版/データベースプレゼンテーションは，Read only の出版を行うのみであるのに対し，提案手法は form（3.3.3 項）による Weite 機能も行える点において優位性がある.

7.4.4 独立型 SuperSQL との比較

独立型 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 (4.4 節) により，必要最小限のファイル生成や，生成した Web コンテンツのメンテナンス性の向上，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，独立型 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，6.4 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が優っている.

第 8 章

結論

8.1 まとめ

本研究では，HTML, PHP 上での SuperSQL を利用した宣言的なクエリによる DB プレゼンテーション・入力 (form) 生成を実現した．開発コスト（コード行数）は世の中の 79%の Web 開発で使われている手続き型言語 PHP と比較して 14〜35%となった．実装では，実行分岐処理，及びデータ, 構造, スタイルの分離機構により，SuperSQL の実行におけるサーバー負荷を軽減を実現した．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, データ更新なしのケースでは, 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することができる結果となり, 有用性を示した. 本手法を用いた Web ページ生成は，DB 問い合わせ結果が同じ場合、タプル数が増えるほど有用であるという結果となった．また，公開されている Web テンプレートに対する埋め込みは、146 件中 99%で可となった．

宣言的な一つの SuperSQL クエリ（ワンソース）によるレスポンシブデザイン生成では，端末の画面横幅ごとのレスポンシブデザインの指定を装飾子を用いて宣言的に記述する指定生成と，一つの記述レイアウトをベースにして自動でレスポンシブデザインを生成する自動生成の 2 つの生成機構を実現した．また，自動生成の評価では，自動生成した場合と指定生成（手動でレスポンシブデザイン用の装飾子を指定してレスポンシブデザインを作成）した場合を比較して，約 7 割のレスポンシブ指定が一致する結果となり，有用性を示した.

8.2 今後の課題

今後の課題としては下記が挙げられる.

- クライアントサイドへキャッシュする機構の検討
- レスポンシブデザインの配置決定処理における処理速度向上に関する検討
- 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対応の検討

8.3 未来像

今回の提案手法のコード量評価では，比較的単純なデータベースプレゼンテーションの表が実装対象となっていた．SuperSQL では，図 8.1 に示すような複雑な表，クロス表，二次元可視化などの生成も宣言的な短いクエリにより実現可能である．手続き

図 8.1: 埋め込み型 SuperSQL による高度な DB プレゼンテーション

型言語を用いてこれらを実装する場合，相当量のコードを記述する必要があると考えられる．世の中のこれらのデータベースプレゼンテーションを SuperSQL で実装すれば，開発コストは減り，より高度なデータベースプレゼンテーションの実現が可能となる．

また，SuperSQL は宣言的な記述による図 8.1 の右下のような三次元可視化の生成も可能である．今回は Web アプリケーション開発に焦点を当て，手続き型言語 PHP を対象にした提案を行ったが，Web にとらわれず，例えば Unity のコード上へ SuperSQL の埋め込みを適用すると，更に高度なデータベースプレゼンテーションを低コストで実現可能であると考えられる．

”宣言的なより少ない記述による，より高度なデータベースプレゼンテーションを”

これが我々SuperSQL 研究者の求める未来へのスローガンである．

付 録 A

SuperSQLの埋め込み評価で使用したWebテンプレート

AA　シンプルテンプレート

tp_bar2_black https://template-party.com/template/tp_bar2/tp_bar2_red/

tp_biz24_white01 https://template-party.com/template/tp_biz24/tp_biz24_white01/

tp_biz25_black https://template-party.com/template/tp_biz25/tp_biz25_black/

tp_biz27_brown https://template-party.com/template/tp_biz27/tp_biz27_brown/

tp_biz28_blue https://template-party.com/template/tp_biz28/tp_biz28_blue

tp_biz29_pink https://template-party.com/template/tp_biz29/tp_biz29_pink

tp_biz30_blue https://template-party.com/template/tp_biz30/tp_biz30_blue

tp_biz31_red https://template-party.com/template/tp_biz31/tp_biz31_red

tp_biz32_orange https://template-party.com/template/tp_biz32/tp_biz32_orange

tp_biz33_green01 https://template-party.com/template/tp_biz33/tp_biz33_green01

tp_cafe11_skyblue https://template-party.com/template/tp_cafe11/tp_cafe11_skyblue/

tp_cafe12_wood_orange https://template-party.com/template/tp_cafe12/tp_cafe12_wood_orange/

tp_car7_black_blue https://template-party.com/template/tp_cafe17/tp_car7_black_blue/

tp_esthe2/tp_esthe2_brown https://template-party.com/template/tp_esthe2/tp_esthe2_brown/

tp_foods6_black_green https://template-party.com/template/tp_foods6/tp_foods6_black_green/

34 件 https://www.coolwebwindow.com/

18 件 http://www.designnow.net/

AB　フレームワークを使用したテンプレート

AB.1　Bootstrap

booster https://freehtml5.co/booster-free-html5-bootstrap-template/

Build https://freehtml5.co/build-free-html5-bootstrap-template/

Clean https://freehtml5.co/clean-free-html5-bootstrap-template/

Crew https://freehtml5.co/crew-free-html5-bootstrap-template/

Display https://freehtml5.co/display-free-html5-template-using-bootstrap/

Hydrogen https://freehtml5.co/hydrogen-free-html5-bootstrap-template/

Light https://freehtml5.co/light-free-html5-template-bootstrap/

Outline https://freehtml5.co/outline-free-html5-bootstrap-template/

relic https://freehtml5.co/relic-free-html5-template-using-bootstrap/

Single https://freehtml5.co/single-free-html5-bootstrap-template/

Slant https://freehtml5.co/single-free-html5-bootstrap-template/

Sprint https://freehtml5.co/sprint-free-html5-template-bootstrap/

Verge https://freehtml5.co/verge-free-html5-bootstrap-template/

Work https://freehtml5.co/work-free-html5-template-bootstrap/

Elegant Restaurant Website https://www.templateshub.net/template/Elegant-Restaurant-Website

AB.2　Material Design Lite

android-dot-com https://getmdl.io/templates/android-dot-com/index.html

article https://getmdl.io/templates/article/index.html

blog https://getmdl.io/templates/blog/index.html

dashboard https://getmdl.io/templates/dashboard/index.html

mdl-templates https://getmdl.io/templates/index.html

text-only https://getmdl.io/templates/text-only/index.html

AB.3 Pure

pure-layout-blog https://purecss.io/layouts/blog/

pure-layout-email https://purecss.io/layouts/email/

pure-layout-gallery https://purecss.io/layouts/gallery/

pure-layout-marketing https://purecss.io/layouts/marketing/

pure-layout-pricing https://purecss.io/layouts/pricing/

pure-layout-side-menu https://purecss.io/layouts/side-menu/

pure-layout-tucked-menu-vertical https://purecss.io/layouts/tucked-menu-vertical/

pure-layout-tucked-menu https://purecss.io/layouts/tucked-menu/

AB.4 Foundation

8 件 https://get.foundation/templates.html

16 件 https://get.foundation/templates-f5.html

AB.5 SkyBlue CSS

eSMITHY https://esmithy.net/

HaciendaRodizio http://haciendarodizio.com

NullObject http://null-object.com

skyblue-gh-pages https://muffinman.io/skyblue/

TeslaSmartHouse http://teslasmarthouse.com/

AB.6 HTML5 Boilerplate

Mattias Hagberg ∞ Frontend Developer https://www.mattias.pw/

Lancaster EMS https://www.lemsa.com/

Paul O ' Rely https://orely.me/

AB.7 UIkit

Home https://smokeyfro.com/themes

kreativan https://www.kreativan.net/

Layout http://layoutnet.com.br/

rmcreations https://www.robbertmastebroek.com/

sports https://simposiocelafiscs.org.br

AB.8 その他フレームワーク

abele https://www.os-templates.com/free-website-templates/abele

colossus https://www.os-templates.com/free-website-templates/colossus

viral https://www.os-templates.com/free-website-templates/viral

cascadeframework-master http://jslegers.github.io/cascadeframework/

century-master https://githubprofile.github.io/century/

Extant http://demo.cssmoban.com/cssthemes6/h5lt_2_Extant/index.html

html5up-highlights https://html5up.net/highlights

html5up-landed https://html5up.net/landed

html5up-photon https://html5up.net/photon

jquery.columns-master http://elclanrs.github.io/jquery.columns/

mrmrs-mnml-f6eb70d http://mrmrs.cc/mnml/

pack186-caprice-free-html-template http://elemisfreebies.com/ed/demos/caprice/

web https://p.w3layouts.com/demos/fixduty/web/

web-1 https://demo.w3layouts.com/demos_new/template_demo/24-01-2018/harbour-demo_Free/1069599531/web/index.html

production https://colorlib.com/polygon/gentelella/

Sedna https://tympanus.net/Freebies/Sedna/

singolo-template https://www.psdchat.com/resources/templates/psd-templates/singolo/

emplated-horizons https://templated.co/horizons

templated-ion https://templated.co/ion

templated-linear https://templated.co/linear

templated-transit https://templated.co/transit

templatemo_406_flex https://templatemo.com/live/templatemo_406_flex

templatemo_434_masonry https://templatemo.com/live/templatemo_434_masonry

templatemo_441_volton https://templatemo.com/live/templatemo_441_volton

Varna-Template-master https://github.com/drac/Varna-Template

参考文献

[1] 大規模なOSS「写真共有・管理アプリ Gallery」. http://galleryproject.org/.

[2] Community Judge! (2018 年度 慶應義塾大学理工学部情報工学実験最優秀賞). http://sun03.std.expr.st.keio.ac.jp/~j161691k/webapp/index.php.

[3] コンサートグッズを代行購入するサイト「コンサートグッズ代行サイト」. http://sun03.std.expr.st.keio.ac.jp/~j164001s/webapp/index.php.

[4] 「チケット出品サイト」. http://sun03.std.expr.st.keio.ac.jp/~j170017h/webapp/index.php.

[5] パフォーマーとイベントをつなぐマッチングサイト「パフォマ」. http://sun03.std.expr.st.keio.ac.jp/~j170549k/webapp/index.php.

[6] 就職活動支援サイト「Karriere」. http://sun03.std.expr.st.keio.ac.jp/~j172055y/webapp/index.php.

[7] W3Techs. http://w3techs.com/. Usage Statistics and Market Share of PHP for Websites.

[8] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, Vol. 26, No. 1, pp. 64–69, January 1983.

[9] Motomichi Toyama. Supersql: An extended sql for database publishing and presentation. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, SIGMOD '98, pp. 584–586, New York, NY, USA, 1998. Association for Computing Machinery.

[10] 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.

[11] SuperSQL. http://ssql.db.ics.keio.ac.jp.

[12] Kento Goto and Motomichi Toyama. Mobile Web Application Generation Features For SuperSQL. In *Proceedings of the 20th International Database Engineering & Applications Symposium*, IDEAS 2016, pp. 308–315, 2016.

[13] Ethan Marcotte. *A List Apart: Responsive Web Design*. Eyrolles, 2010.

[14] Are you ready for Google ' s mobile algorithm change? `https://thenextweb.com/google/2015/04/22/are-you-ready-for-googles-mobile-algorithm-change/`.

[15] Proof that no ranking boost for responsive sites exists in 2017. `https://searchengineland.com/proof-no-ranking-boost-responsive-sites-exists-2017-278886`.

[16] Responsive Templates from ThemeForest. `https://themeforest.net/tags/responsive`.

[17] Bootstrap: The world's most popular mobile-first and responsive front-end framework. `http://getbootstrap.com/`.

[18] Amazon.com: Online Shopping for Electronics, Apparel, Computers, Books, DVDs & more. `https://www.amazon.com/`.

[19] Google. `https://www.google.com/`.

[20] Yahoo!ニュース. `https://news.yahoo.co.jp/`.

[21] Toshiyuki Seto, Takuhiro Nagafuji, and Motomichi Toyama. Generating html sources with tfe enhanced sql. In *Proceedings of the 1997 ACM Symposium on Applied Computing*, SAC '97, pp. 96–100, New York, NY, USA, 1997. ACM.

[22] 有薗チエ. Web アプリケーション開発のための SuperSQL の拡張. 慶應義塾大学開放環境科学修士論文, 2010.

[23] Selenium WebDriver. `http://www.seleniumhq.org/projects/webdriver/`.

[24] Michael O. Leavitt & Ben Shneiderman. *Research-Based Web Design & Usability Guidelines*. U.S. Deptartment of Health and Human Ser; Enlarged/Expanded edition, 2006.

[25] Deborah J. Mayhew. *Principles and Guidelines in User Interface Design*. Prentice Hall, 1992.

[26] A Nygren, E. & Allard. Between the clicks: Skilled users scanning of pages. In *Proceedings of the 2nd Conference on Human Factors and the Web*, 1996.

[27] J.N Smith, S.L. & Mosier. Guidelines for designing user interface software. In *The MITRE Corporation Technical Report*, 1986.

[28] J.N Smith, S.L. & Mosier. Predicting the usability of alphanumeric displays. In *Doctoral Dissertation, Houston, TX: Rice University*, 1984.

[29] P Wright. Presenting technical information: A survey of research findings. In *Instructional Science, 6, pp. 93-134*, 1977.

[30] CINEMA WORLD FREE CSS TEMPLATE. http://www.free-css.com/free-css-templates/page138/cinema-world.

[31] PHP.net. `http://php.net/`.

[32] Java. https://www.java.com/ja/.

[33] Ruby on Rails — A web-application framework that includes everything needed to create database-backed web applications according to the Model-View-Controller (MVC) pattern. `https://rubyonrails.org/`.

[34] ASP.NET. http://www.asp.net/.

[35] JavaScript. https://developer.mozilla.org/ja/docs/Web/JavaScript.

[36] jQuery. http://jquery.com/.

[37] Ruby: A Programmer's Best Friend. `http://www.ruby-lang.org/`.

[38] The Perl Programming Language. `https://www.perl.org/`.

[39] CakePHP: The rapid development php framework. `http://cakephp.org/`.

[40] Laravel - The PHP Framework For Web Artisans. `https://laravel.com/`.

[41] rainner/biscuit-php. `https://packagist.org/packages/rainner/biscuit-php`.

[42] Symfony, High Performance PHP Framework for Web Development. `https://symfony.com/`.

[43] Play Framework - Build Modern and Scalable Web Apps with Java and Scala. https://www.playframework.com/.

[44] Apache Tapestry Home Page. http://tapestry.apache.org/.

[45] The Apache Velocity Project. http://velocity.apache.org/.

[46] Ark. https://metacpan.org/release/Ark.

[47] Catalyst — Perl MVC web application framework. http://www.catalystframework.org/.

[48] Mojolicious - Perl real-time web framework. http://mojolicious.org/.

[49] Django: The Web framework for perfectionists with deadlines. https://www.djangoproject.com/.

[50] TurboGears. http://turbogears.org/.

[51] Flask (A Python Microframework). http://flask.pocoo.org/.

[52] Design Framework technologies Web Usage Statistics. https://trends.builtwith.com/docinfo/design-framework.

[53] Love it or Hate it, Bootstrap is Winning the Web. https://www.ostraining.com/blog/coding/bootstrap-winning/.

[54] Foundation — The most advanced responsive front-end framework in the world. http://foundation.zurb.com/.

[55] Material Design Lite. https://getmdl.io/.

[56] Souer, Jurriaan, and et al. Situational requirements engineering for the development of content management system-based web applications. *International Journal of Web Engineering and Technology*, pp. 420–440, 2007.

[57] Jurriaan Souer, Lutzen Luinenburg, Johan Versendaal, Inge van de Weerd, and Sjaak Brinkkemper. Engineering a design method for web content management implementations. In *Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services*, iiWAS '08, pp. 351–358, New York, NY, USA, 2008. ACM.

[58] Blog Tool, Publishing Platform, and CMS - WordPress. https://wordpress.org/.

[59] Joomla! The CMS Trusted By Millions for their Websites. https://www.joomla.org/.

[60] Drupal - Open Source CMS — Drupal.org. https://www.drupal.org/.

[61] Jochen Rode, Yogita Bhardwaj, Manuel A Pérez-Quiñones, Mary Beth Rosson, and Jonathan Howarth. As easy as “click”: End-user web engineering. In *Proceedings of the Web Engineering*, pp. 478–488. Springer, 2005.

[62] Kim Marriott, Peter Moulder, and Nathan Hurst. Html automatic table layout. *ACM Trans. Web*, Vol. 7, No. 1, pp. 4:1–4:27, March 2013.

[63] Hesam Samimi, Max Schäfer, Shay Artzi, Todd Millstein, Frank Tip, and Laurie Hendren. Automated repair of html generation errors in php applications using string constraint solving. In *Proceedings of the 34th International Conference on Software Engineering*, pp. 277–287. IEEE Press, 2012.

[64] Stphane Ducasse, Adrian Lienhard, and Lukas Renggli. Seaside: A flexible environment for building dynamic web applications. *Software, IEEE*, Vol. 24, pp. 56–63, 10 2007.

[65] Shiro Udoguchi, Tadashi Iijima, and Motomichi Toyama. Application of SuperSQL query language for the migration from a relational to an object-oriented database. In *Proceedings of the International Database Engineering and Applications Symposium*, pp. 207–216, 2000.

[66] 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（TOD）, Vol. 42, No. SIG08(TOD10), pp. 66–95, jul 2001.

[67] 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（TOD）, Vol. 46, No. SIG13(TOD27), pp. 65–77, sep 2005.

[68] 有澤達也, 恭子石川, 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.

[69] 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（TOD）, Vol. 46, No. SIG13(TOD27), pp. 1–15, sep 2005.

[70] Zhe Jin and Motomichi Toyama. A Prototype Implementation of PPX: Pretty Printer for XML. In *Fourth International Conference on Internet and Web Applications and Services*, pp. 149–156, 2009.

[71] 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（TOD）, Vol. 3, No. 4, pp. 13–33, dec 2010.

[72] Yoko Maeda and Motomichi Toyama. ACTIVIEW: Adaptive data presentation using SuperSQL. In *Proceedings of the 27th International Conference on Very Large Data Bases*, pp. 695–696, 2001.

[73] Sang-Gyu Shin, Maeda Yoko, and Motomichi Toyama. ACTIVIEW: Implementation of Adaptive Web View Using SuperSQL. In *The 6th International Conference on Informatics and Systems*, pp. 25–34, 2008.

[74] 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（TOD）, Vol. 2, No. 3, pp. 112–129, sep 2009.

[75] Kento Goto, Ryosuke Koshijima, and Motomichi Toyama. Generating Desktop and Mobile Web Pages from a Single SuperSQL Query. In *Proceedings of the 19th International Database Engineering & Applications Symposium*, pp. 222–223, 2015.

[76] Tai-Suk Kim, Sang-Gyu Shin, and Motomichi Toyama. Efficient Media Publication Using SuperSQL Processor. *Journal of Korea Society for Simulation*, Vol. 15, No. 1, pp. 59–67, 2006.

[77] Sang-Gyu Shin, Tai-Suk Kim, and Motomichi Toyama. 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.

[78] Ria Mae Borromeo and Motomichi Toyama. Optimization of SuperSQL Execution by Query Decomposition. In *Proceedings of the Fifth International Conference on Advances in Databases, Knowledge, and Data Applications*, pp. 65–70, 2013.

[79] Arnaud Wolf, Ria Mae Borromeo, and Motomichi Toyama. A Data Retrieval Model Based on Independence Rules for SuperSQL. In *Proceedings of the 19th International Database Engineering & Applications Symposium*, pp. 208–209, 2015.

[80] Masato Kiya, Kento Goto, and Motomichi Toyama. GENERATE eHTML: Embedding SuperSQL Queries in HTML. In *Proceedings of the 19th International Database Engineering & Applications Symposium*, pp. 224–225, 2015.

[81] Aaron Koehl and Haining Wang. M.site: Efficient content adaptation for mobile devices. In *Proceedings of the 13th International Middleware Conference*, Middleware '12, pp. 41–60, New York, NY, USA, 2012. Springer-Verlag New York, Inc.

[82] Nishant Sinha and Rezwana Karim. Compiling mockups to flexible uis. In *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2013, pp. 312–322, New York, NY, USA, 2013. ACM.

[83] Nishant Sinha and Rezwana Karim. Responsive designs in a snap. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2015, pp. 544–554, New York, NY, USA, 2015. ACM.

[84] Adobe FrameMaker. `https://www.adobe.com/products/framemaker.html`.

[85] Adobe InDesign. `https://www.adobe.com/products/indesign.html`.

[86] Datalogics Pager. `https://www.datalogics.com/`.

[87] QuarkXPress — The fully-integrated graphic design and layout software for professional print and digital production. `http://www.quark.com/Products/QuarkXPress/`.

[88] Xyvision. `http://xml.coverpages.org/xyvisionLivelink.html`.

[89] Arbortext Advanced Print Publisher - Single-Sourcing Solutions. `https://www.single-sourcing.com/project/arbortext-layout-developer/`.

[90] priint:suite The leading multichannel publishing platform for product communication. `https://priint.com/`.

[91] Jacci Howard Bear. *What's Involved in Desktop Publishing?* Lifewire, 2019.

[92] R. Shotwell. *Getting into database publishing.* Publishers Weekly, 1981.

[93] Joyce Duncan Falk. The historian enters the electronic age: Bibliographical and database publishing. *JSTOR*, Vol. 4, No. 2, pp. 35–42, 1982.

[94] S. K Sieck. *Database publishing applications of optical videodiscs*. Learned Information, 1983.

[95] Jon Bing. *Electronic Publishing-Part One: Database Publishing*. HeinOnline, 1987.

[96] R. Keith and P.E. Nelms. Database publishing: Applications, examples, and fundamental concepts. *Computers & Industrial Engineering*, Vol. 17, No. 14, pp. 390–396, 1989.

[97] Michael O. Petrea and Kent M. Taylor. Multimedia publishing from a database. *AT&T Technical Journal*, Vol. 68, No. 4, pp. 61–71, 1989.

[98] Jeffrey McArthur. Developing Database Publishing Systems Using TEX. In *TEXNorthEast Conference*, pp. 188–194, 1998.

[99] Renée J. Miller, Odysseas G. Tsatalos, and John H. Williams. Dataweb: Customizable database publishing for the web. *IEEE MultiMedia*, Vol. 4, pp. 14–21, 1997.

[100] Michael Carey, Daniela Florescu, Zachary Ives, Ying Lu, Jayavel Shanmugasundaram, Eugene Shekita, and Subbu Subramanian. Xperanto: Publishing object-relational data as xml. In *In WebDB*, pp. 105–110, 2000.

[101] Michael Carey, Jerry Kiernan, Jayavel Shanmugasundaram, Eugene Shekita, and Subbu Subramanian. Xperanto: A middleware for publishing object-relational data as xml documents. *Proceedings of the 26th International Conference on Very Large Data Bases, VLDB'00*, 12 2000.

[102] Alin Deutsch and Yannis Papakonstantinou. Privacy in Database Publishing. In *International Conference on Database Theory, ICDT 2005*, pp. 230–245, 2005.

[103] Alin Deutsch. *Privacy in database publishing: a bayesian perspective*. Springer, 2008.

[104] Millist Vincent, Mukesh Mohania, and Mizuho Iwaihara. Detecting privacy violations in database publishing using disjoint queries. In *EDBT 2009 : 12th International Conference on Extending Database Technology*, pp. 252–262, January 2009.

[105] Deming Dou and Stphane Coulondre. Disclosure detection over data streams in database publishing. In *ACM International Conference Proceeding Series*, pp. 8–13, March 2011.

[106] J. Mokadam1 Pragati and S.T.Singh. Data attribute security and privacy in collaborative distributed database publishing. *International Journal of Engineering Inventions*, Vol. 3, No. 12, pp. 60–65, 2014.