SuperSQLによるストリームデータに対応した動的ページ生成の実装

照井　恵太† 　五嶋　研人† 　遠山　元道†

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

あらまし　SuperSQL は，独自のクエリを記述することによって関係データベースの出力結果を構造化し，多様なレイアウト表現を可能とする SQL の拡張言語である．従来の SuperSQL ではデータを一括して取得・提示を行う Web ページの生成は実現していたが，この方法はデータベース内の情報が SuperSQL 実行時のものを参照するためストリームデータのような更新頻度が非常に高いデータに対応することが難しい．そこで本研究ではストリーム処理を行うことのできる DBMS である PipelineDB と SuperSQL を用いて，更新頻度が非常に高いデータに対して最新の情報を参照して非同期にページを更新するような Web ページの生成機能の実装を提案する．これによって株価一覧を表示するページ中に随時株価の最新値を表示するような Web ページの作成が可能になる．

キーワード　データベース，SQL，SuperSQL，ストリームデータ，HTML

1　はじめに

SuperSQL は，独自のクエリを記述することによって関係データベースの出力結果を構造化し，多様なレイアウト表現を可能とする SQL の拡張言語である．通常の SQL では，シンプルでフラットな表しか再現できないが，SuperSQL を用いることで様々な表を作成することができる．また，SuperSQL を用いることによって，従来の方法に比べてはるかに少ないコードで Web ページや php などを生成することができる．

上記の SuperSQL ではクエリに記述された属性のデータをすべて一括して取得して Web ページを生成するためデータは SuperSQL 実行時のデータベースの情報が参照され，ストリームデータのような更新頻度が非常に高いデータに対応することが難しかった．そこで本研究では SuperSQL を用いて，更新頻度が非常に高いデータに対して最新の情報を参照して非同期にページを更新するような Web ページの生成機能を実装し，使用する DBMS を PipelineDB というストリーム処理に対応した DBMS を使用することによって，より効果的な Web ページが生成されることを目的としている．提案実現のため，SuperSQL クエリにおいて非同期に Web ページが更新されるスクリプトを生成するための新たなレイアウト構造生成機能を開発した他，従来の SuperSQL では実現できていなかったストリームデータに対応した Web ページの生成機構の拡張を行った．

また，本研究手法では非同期にデータを取得する上でブラウザの技術対応とリアルタイム性が求められるが，これらは独立した要求である．そこでデータの取得方法に Ajax を用いた技術と Server Sent Events を用いた 2 つの方法で非同期な Web ページの生成を実現している．ユーザが SuperSQL を用いて非同期な Web ページの生成を行う際，ブラウザの技術対応とリアルタイム性を求める場合とでデータの取得方法を選択できる機能を提案している．

以降，本稿では第 2 章で SuperSQL について，第 3 章では関連技術，関連研究について，第 4 章では SuperSQL によるストリームデータに対応した動的 Web ページの生成について，第 5 章では実験・評価について述べ，第 6 章ではまとめを記述する．

2　SuperSQL とは

この章では，SuperSQL について簡単に述べる．図 1 に SuperSQL のアーキテクチャを示す．

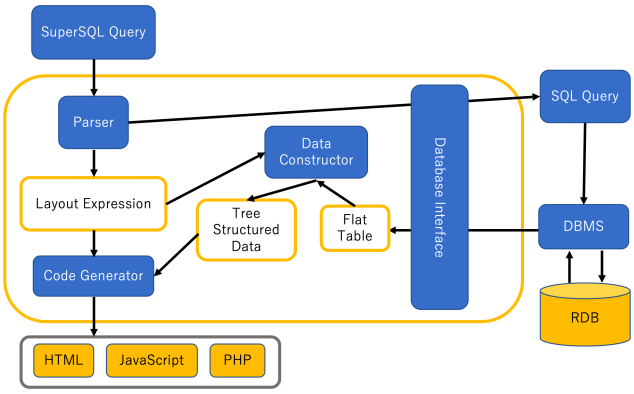

図 1　SuperSQL のアーキテクチャ

SuperSQL は関係データベースの出力結果を構造化し，多様なレイアウト表現を可能とする SQL の拡張言語であり，慶應義塾大学遠山研究室で開発されている [1], [2]．そのクエリは SQL の SELECT 句を GENERATE <*media*> <*TFE*> の構文を持つ GENERATE 句で置き換えたものである．ここで <*media*> は出力媒体を示し，HTML，PDF，Mobile_HTML5 [3] などの指定ができる．また <*TFE*> はターゲットリストの拡張である Target Form Expression を表し，結合子，反復子などのレイアウト指定演算子を持つ一種の式である．

2.1 結　合　子

結合子はデータベースから得られたデータをどの方向 (次元) に結合するかを指定する演算子であり，以下の 3 種類がある．括弧内はクエリ中の演算子を示している．

- 水平結合子 (,)

データを横に結合して出力する．

例：name, place

name	place

- 垂直結合子 (!)

データを縦に結合して出力する．

例：name! place

name
place

2.2 反　復　子

反復子は指定する方向に，データベースの値があるだけ繰り返して表示する．また反復子はただ構造を指定するだけではなく，そのネストの関係によって属性間の関連を指定できる．例えば

[部署名]! , [雇用者名]! , [給料]!

とした場合には各属性間に関係はなく，単に各々の一覧が表示されるだけである．一方，ネストを利用して

[部署名! [雇用者名, 給料]!]!

とした場合には，その部署毎に雇用者名と給料の一覧が表示されるといったように，属性間の関連が指定される．以下，その種類について述べる．

- 水平反復子 ([ ],)

データインスタンスがある限り，その属性のデータを横に繰り返し表示する．

例：[Name],

name1	name2	・・・	name10

- 垂直反復子 ([ ]!)

データインスタンスがある限り，その属性のデータを縦に繰り返し表示する．

例：[Name]!

name1
name2
・・・
name10

2.3 装　飾　子

SuperSQL では関係データベースより抽出された情報に，文字サイズ，文字スタイル，横幅，文字色，背景，高さ，位置などの情報を付加できる．これらは装飾演算子 (@) によって指定する．

< 属性名 >@{< 装飾指定 >}

装飾指定は “装飾子の名称 = その内容” として指定する．複数指定するときは各々を “,” で区切る．

[name@{width=100, color=red}]!

3 関連技術，関連研究

3.1 関 連 技 術

3.1.1 PipelineDB

PipelineDB [4] とはオープンソースのストリーミング SQL リレーショナルデータベースである．PostgreSQL と互換性があり，PostgreSQL 互換のクエリでストリームデータに問い合わせることが可能である．システム上に大量に流れてくるデータを PipelineDB 上に流すことで，一定期間分のデータに対する SQL 問い合わせを継続的に実行することを実現する．この問い合わせを継続的クエリといい，PostgreSQL のような静的データを扱うシステムではデータを溜め込み，1 度クエリを発行すると結果が 1 度だけ返される．対して PipelineDB はクエリをあらかじめ用意し，データを流すたびに結果を返す．データを流す部分は Stream というテーブルに似た宣言方法のものを用い，INSERT 文で Stream にデータを流すことができる．Stream では流れてきたデータ自体は保持せず，Stream を参照元とする Continous View という SQL ビューに似た仕組みを通じてストリームデータの集計結果のみ継続的に更新し続け保持するためデータベースとしての領域は多く必要としない．ストリームデータの導入から集計までを SQL クエリで完結させることができることから，データの抽出，整形，導入といった ETL の作業を簡素にすることが可能である．

ストリームデータ処理基盤には Apache Kafka [5]，Spark Streaming [7]，Amazon Kinesis [6] などが挙げられるが，SuperSQL では以前から DBMS に PostgreSQL が代表的に使用されており，PostgreSQL との接続の基盤がすでに存在するため本稿では PipelineDB を使用した．

3.2 非同期通信によるデータ取得

本稿の非同期通信は，google map の登場から注目を集めてきた JavaScript と DOM を用いた非同期通信を行う Ajax と，W3C で提案されている html5 関連 API の一種である Server Sent Events を基盤として実現がなされている．

Ajax を用いた非同期にデータを更新する方法としては，polling というクラインアントから最新のデータがあるか，設定可能な一定の間隔でサーバをチェックする．プルの頻度は高精度のデータを保証するために多くなければならないが，頻度の多さは冗長なチェックを生じネットワークトラフィックを増やすことになりかねない。一方、プルの頻度の少なさは、更新を見逃す結果になる可能性がある．

Server Sent Events を用いた非同期にデータを更新する方法としては，HTTP server push というデータをサーバからクライアントにデータを転送する．その際，サーバはレスポンスデータをクライアントに返した後も接続を切断しない．この特

徴により，クライアントはサーバから断続的に送信されてくるデータを随時受け取ることができ，クライアントとサーバの再接続のコストがかからない．

4　SuperSQL による動的ページ生成機能

ここでは，本提案手法によるストリームデータに対応した動的 Web ページの生成について述べる．

4.1　クエリによる非同期更新部分の指定

本提案手法では，SuperSQL の反復子に対し装飾し @{stream=n} を指定することにより，右辺に指定された指定時間で非同期に更新を行う Web ページの生成を行う．n は数値であり，n ミリ秒で更新するという意味である．n を指定しない場合は 1000 ミリ秒で更新が行われる．以下に SuperSQL の装飾子として実装した宣言を示す．

stream 装飾子の宣言

```
[ TEF1, [ TEF2 ]! ]!@{stream=n}
```

4.2　アーキテクチャ

図 2 に提案手法のアーキテクチャを示す．ユーザは SuperSQL クエリを実行することによって HTML, CSS, JavaScript, PHP が生成される．

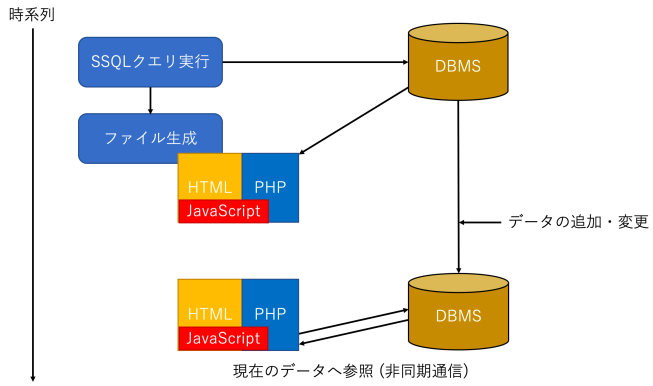

図 2　提案手法のアーキテクチャ

PHP が実行されることによってデータベースにデータの問い合わせが行われ，JSON 形式のデータが生成される．閲覧者には HTML と JavaScript がデータを受け取ったものが表示される．

4.3　データの取得機構

提案手法では 3.2 で述べたように，2 つのデータの取得手法がある．ここでは，本稿で提案する 2 種類のデータ取得手法について述べる．

表 1 に 2 種類のデータ取得手法の概要を示す．pull モードは，装飾子 stream と一緒に “mode=‘pull’ ” の記述を行うことによって指定することが可能であり，mode 指定がなかった場合も pull モードが使用される．pull モードでは，Ajax を用いたクライアントからサーバに向けてデータのリクエストが繰り返し送られ，その都度サーバはクライアントにデータ送信するため，通信コストが高く，リアルタイム性が低い一方でブラウザの互換性は高く，閲覧者のブラウザの種類・バージョンを考慮する必要性が低いという特徴がある．図 3 に pull モードにおけるアーキテクチャを示す．

表 1　データ取得方法

	pull モード	push モード
採用技術	Ajax	Server Sent Events
通信コスト	高い	低い
ブラウザ互換性	高い	低い
リアルタイム性	低い	高い

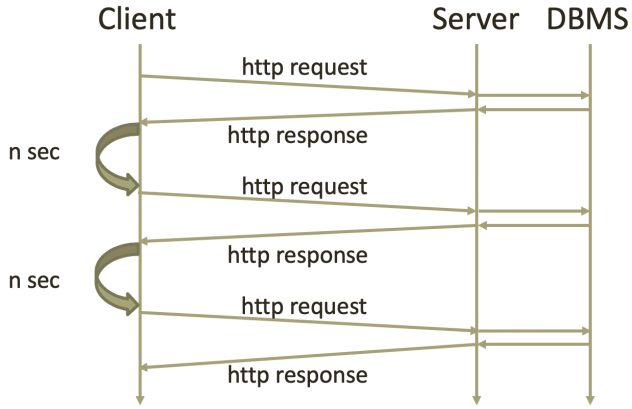

図 3　pull モードにおけるアーキテクチャ

push モードでは，装飾子 stream と一緒に “mode=‘push’ ” の記述を行うことによって指定することができる．push モードでは Server Sent Events を用いた一度クライアントとサーバーのコネクションが確立されればサーバからクライアントに向けて一方的にではあるがリクエストを必要としないデータ送信が可能であり，コネクションの確立のやり直しが無い分通信コストが低く，リアルタイム性が高い一方で古いブラウザでは動作できないという問題点がある．図 4 に push モードにおけるアーキテクチャを示す．

ここで説明の為，仮想通貨と特定の仮想通貨のレートを監視したいユーザを対象としたデータベースを例に挙げる．例で使うデータベースは以下の通りである．

- crypto(id，exchange，name，rate)
- users(id，name)
- monitor(c_id, u_id)

crypto は 3.1.1 で述べた Continuos View であり，ある取引所で取り扱う仮想通貨の現在の日本円とのレートである．users はある仮想通貨のレートを監視する人のテーブルである．monitor は crypto と users との関係を表す中間テーブルである．crypto の参照元である Stream には常にデータ各仮想通貨のレートが INSERT され続けている．

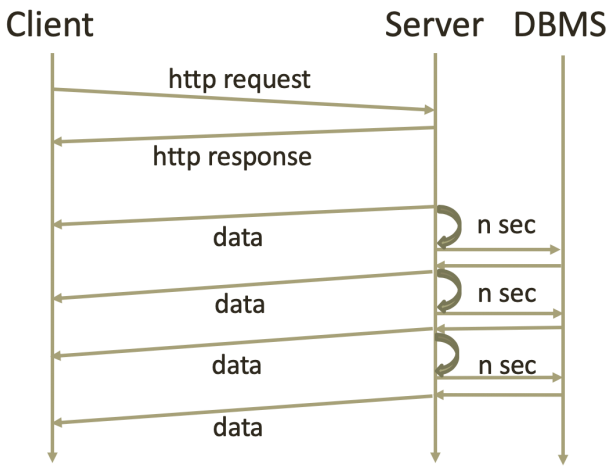

図 4　push モードにおけるアーキテクチャ

sample.ssql

```
GENERATE HTML_bootstrap
 [ null((asc)u.id), u.name,
    [ c.exchange, c.name, c.rate
    ]!
]!@{stream=3000}
FROM users u, crypto c, monitor m
WHERE c.id = m.c_id AND u.id = m.u_id
```

sample.ssql を実行することにより，図 5 のような結果が得られる．

User1	Zaif	BCH	17005
	Zaif	BTC	433050
User2	bitflyer	BTC	435001.4
User3	Zaif	BTC	433050
	Zaif	MONA	134
	bitflyer	BTC	435001.4
User4	bitflyer	BTC	435001.4
User5	Zaif	BCH	17005
	Zaif	BTC	433050
	Zaif	MONA	134
	Zaif	XEM	6.8601
	bitflyer	BTC	435001.4
	bitflyer	FX_BTC	435084.9

図 5　sample.ssql を実行した生成結果

図 5 において青枠で囲われた部分が非同期に更新される．

5　実験・評価

提案システムの有用性を評価するために以下の評価実験を行った．

- 更新頻度によるクライアントの負荷実験
- サーバ上の PipelineDB における負荷実験

5.1　更新頻度によるクラインアントの負荷実験

この実験では非同期に更新する Web ページの非同期更新時間と閲覧するクライアント側の cpu 使用率による負荷実験を行った．サーバの実験環境は以下の通りである．

- サーバ
- OS: CentOS 7.3.1611
- CPU: Intel(R) Xeon(R) CPU E5-2670 v2 @ 2.50GHz 10 × 2core
- メモリ: 126GB
- DBMS: PipelineDB 0.9.9

また，クライアントの実験環境は以下の通りである．

- クライアント
- OS: Mac OS Mojave
- CPU: 3.1 GHz Intel Core i5
- メモリ: 16GB
- ブラウザ: firefox 64.0

この実験では Web ページの更新頻度を 100, 200, ... , 1000 ミリ秒に設定した生成物を閲覧するクライアントの CPU 使用率の測定を行った．図 6 に更新頻度によるクラインアントの負荷実験の実験結果を表す．

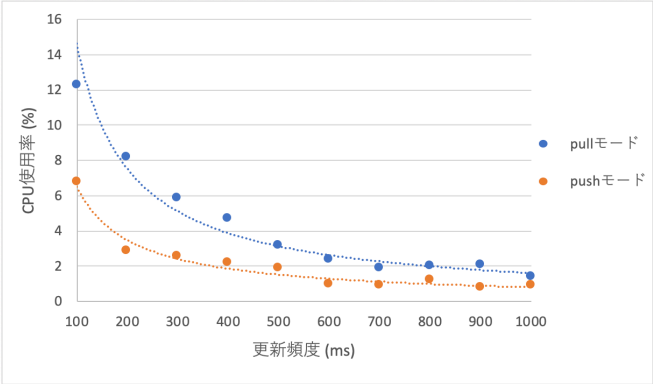

図 6　更新頻度によるクラインアントの負荷実験の実験結果

この結果からクライアント側では pull モードの方が push モードよりも多くの cpu リソースを消費していることがわかる．また，pull モードが pull モードよりも平均で 2.1 倍の CPU リソースを消費していた．これは pull モードではブラウザが更新を行うためにサーバにリクエストを送信し，受け取ったデータから再表示を行うのに対し，push モードではサーバにリクエストを送信する必要がなく送られてきたデータを再表示する操作のみであるからだと考えられる．

5.2　サーバ上の PipelineDB における負荷実験

この実験ではサーバ上の PipelineDB の CPU 使用率から，同じ Web ページにアクセスできるクライアント数を求める実験を行った．サーバ，クライアントの実験環境は 5.1 と同じである．10 ミリ秒で非同期更新する Web ページに 1, 2, ... , 10 件の同時リクエストを送った時のサーバ上の PipelineDB の CPU 使用率を測定する．図 7 にサーバ上の PipelineDB にお

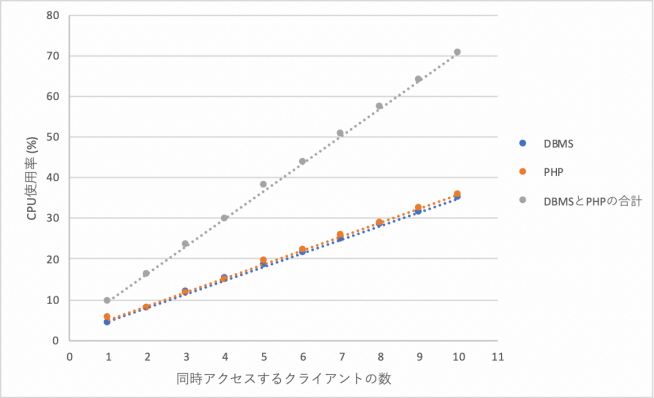

図 7　サーバ上の PipelineDB における負荷実験

ける負荷実験の実験結果を表す．

この結果から同時アクセス数とサーバ上の PipelineDB の CPU 使用率には比例関係がある．この関係から，

$$(\text{CPU 使用率}) \propto (\text{同時アクセス数}) \tag{1}$$

と表せる．また，サーバから見た 1 秒間における同時アクセス数は，

$$(\text{同時アクセス数}) = (\text{実際のクライアント数}) \cdot (\text{PHP のファイル数}) \cdot \frac{(1\text{ 秒})}{(\text{更新間隔})} \tag{2}$$

と表せる．よって本研究の機能を用いた生成物である Web ページに同時にアクセスできる最大クライアント数と，開発者が Web ページに割り当てられる CPU リソースの関係は，

$$(\text{最大クライアント数}) \propto \frac{(\text{利用する CPU 使用率}) \cdot (\text{更新間隔})}{(\text{PHP のファイル数}) \cdot (1\text{ 秒})} \tag{3}$$

となり，サーバマシンのスレッドを考慮すると

$$(\text{最大クライアント数}) \propto \frac{(\text{利用する CPU 使用率}) \cdot (\text{スレッド数}) \cdot (\text{更新間隔})}{(\text{PHP のファイル数}) \cdot (1\text{ 秒})} \tag{4}$$

である．

6　ま　と　め

本稿では，SuperSQL によるストリームデータに対応した動的 Web ページの生成機能を実装した．これにより，非同期更新を行う Web ページを作成するために HTML，JavaScript，PHP など複数のファイルをそれぞれ記述しなければならなかったものが，SuperSQL クエリのみで非同期更新を行う Web ページを作成することを可能にした．

文　献

[1] SuperSQL: http://SuperSQL.db.ics.keio.ac.jp
[2] M. Toyama, “SuperSQL: An Extended SQL for Database Publishing and Presentation”, Proceedings of ACM SIGMOD’98 International Conference on Management of Data, pp. 584-586, 1998.
[3] K. Goto and M. Toyama, “Mobile Web Application Generation Features For SuperSQL”, in Proceedings of the 20th International Database Engineering Applications Symposium, IDEAS 2016, pp. 308-315, 2016.
[4] PipelineDB: https://www.pipelinedb.com/
[5] Jay Kreps, Neha Narkhede, and Jun Rao, “Kafka: a distributed messaging system for log processing”, ACM SIGMOD Workshop on Networking Meets Databases, page 6, 2011.
[6] Amazon Kinesis: http://aws.amazon.com/kinesis/
[7] Spark Streaming: https://spark.apache.org/streaming/