SSFlow: ブロック要素を用いたSuperSQL利用者のためのインタラクティブなNo-Codeツール

栗原　太一† 五嶋　研人† 根本　潤†† 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
†† 慶應義塾大学大学院政策・メディア研究科　〒 252–0882 神奈川県藤沢市遠藤 5322
E-mail: †{kuri,goto}@db.ics.keio.ac.jp, ††nemoto@keio.jp, †††toyama@ics.keio.ac.jp

あらまし　SuperSQL とは SQL の拡張言語であり，独自のクエリを記述して関係データベースの出力結果を構造化することで多彩なレイアウト表現の実現を可能とするものである．本論文では，ブロック要素をドラッグ&ドロップで操作することでキャンバスに追加し，またそのレイアウトを確認することで，SuperSQL での開発を容易にし，コードの修正回数を減らすことを目的とした，No-Code ツール (SSFlow) を提案する．SSFlow は配置されたブロックを元に自動でクエリを生成したり，マウスの操作によって特定の要素を削除する．これにより，利用者はコードを実行して結果を確認し，修正して再度実行するといった負担が低減し，SuperSQL のコード作成にかかる時間も削減する．

キーワード　No-Code，可視化，Block-based，インタラクティブ，SuperSQL

1 はじめに

SuperSQL [1] [2] 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へ問い合わせた結果に対して構造化を行い，出力メディアや多様なレイアウト表現の指定を可能とする SQL の拡張言語である．しかし出力結果が表構造という視覚的なものであるのに対し，その構造を表す SuperSQL クエリは論理的なものであるため，SuperSQL に習熟していない利用者が，自身がイメージしているクエリを作成するまでには時間を要することになる．そこで本論文では，ブロック要素をドラッグ&ドロップで操作することでキャンバスに追加し，またそのレイアウトを確認することで，SuperSQL での開発を容易にし，コードの修正回数を減らすことを目的とした，No-Code ツール (SSFlow) を提案する．SSFlow は配置されたブロックを元に自動でコードを生成したり，マウスの操作によって特定の要素を削除する．これにより，利用者はクエリを実行して結果を確認し，修正して再度実行するといった負担が低減し，SuperSQL のクエリ作成にかかる時間も削減する．以下に本論文の構成を示す．2 章では SuperSQL の概要と SuperSQL クエリ作成における課題を，3 章では SSFlow の概要を，4 章では SSFlow のシステム設計を，5 章では SSFlow のシステムの実装を， 6 章では評価を，7 章ではブロック型のプログラミングツールと No-Code ツールにおける関連研究を，8 章では結論と今後の課題を述べる．

2 SuperSQL とその利用における課題

2.1 SuperSQL の概要

2.1.1 結　合　子

結合子はデータベースから得られたデータをどの方向に結合するかを指定する演算子であり，コンマ (,)，感嘆符 (!)，パーセント記号 (%) の 3 種類がある．属性間をこれらの結合子で区切ることによって，それぞれ水平，垂直，深度方向にレイアウトする． 以下に結合子の使用例を示す．

- 水平結合子 (,)

データを横に結合して出力．
例：Name, Tel

山田花子	03 - 1111 - 2222

- 垂直結合子 (!)

データを縦に結合して出力．
例：Name! Tel

山田花子
03 - 1111 - 2222

- 深度結合子 (%)

データをリンク方向に結合して出力．
例：Name % Tel

山田花子	→	03 - 1111 - 2222

2.1.2 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．一対の角括弧 ([ ]) に上記の結合子を添えたものが反復子となる．以下に反復子の使用例を示す．

- 水平反復子 ([ ],)

インスタンスがある限り，データを横方向に反復して出力．
例：[Name],

name1	name2	...	name10

- 垂直反復子 ([ ]!)

インスタンスがある限り，データを縦方向に反復して出力．

例：[Name]!

name1
name2
...
name10

2.1.3 反復子の入れ子

2.1.2 で述べた反復子をただ並置した場合は、表 1 のように各々の一覧を表示する．

表 1　並置した反復子での表示

駄菓子屋	子供用デニムスーツ	198
子供服	迷路のおもちゃ	800
寝具	300g 入りのあめ	3295
…	…	…

しかしながら，以下の例のように反復子を入れ子にすることで，属性間の関連を指定する．

```
1  GENERATE HTML
2  {
3    [
4      d.name@{width=250}! [i.name@{width=180}, i
           .price@{width=70}]!
5    ]!
6  }@{table}
7  FROM item i, dept d WHERE i.dept = d.id
```

この場合には，表 2 のように売り場ごとの商品の名前と値段の一覧を表示する．

表 2　入れ子にした反復子での表示

駄菓子屋	
300g 入りのあめ	500
600g 入りのあめ	800
子供服	
ベルボトムジーンズ	3000
子供用デニムスーツ	6000
…	…

2.1.4 中括弧 ({ }) によるグルーピング

SuperSQL ではクエリの一部を中括弧 ({ }) で囲むことでその中身を 1 つのかたまりとして扱う．これにより，レイアウト構造が明確になるという利点もある [3].

2.1.5 関　　数

a） image 関数

image 関数を用いると，指定したパスの画像を表示する．image 関数には引数があり，そこにテーブルの属性名及び画像のファイルが存在するディレクトリのパスを指定する．

image(employee.pict, ’./picts’)

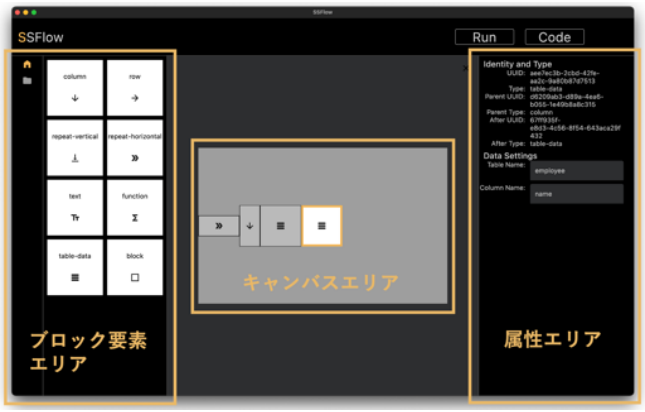

図 1　メイン画面 1

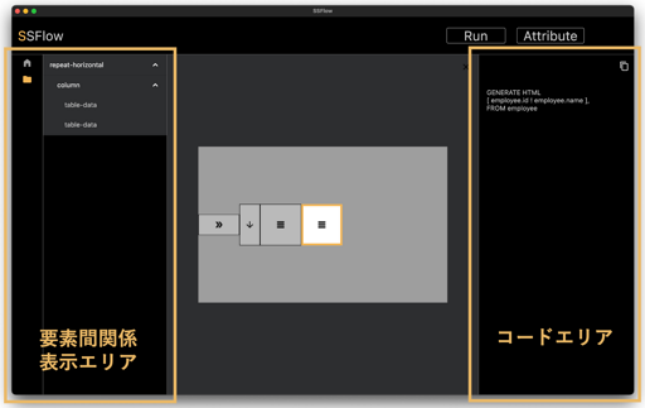

図 2　メイン画面 2

2.2 SuperSQL クエリ作成の課題

現在，SuperSQL クエリ作成では主に SSedit [4] というツールが使用されている．このツールの課題としては，クエリを作成するまでのプロセスとして，まず一度作成してそれを実行し，その実行結果を見てから再度修正するという流れがあるが，このプロセスに時間が掛かり，結果として 1 つのクエリをイメージしてから完成させるまでに時間がかかるということがある．特に複雑なクエリになればなるほど SSedit のように，クエリをテキストで書きながら完成をイメージするということは難しい．また SuperSQL では見た目を整えるために装飾子を用いるが，クエリが複雑になればなるほど装飾子も多くなる [3]．現在主に使われている SSedit のようなツールは，SuperSQL のクエリ生成に習熟している利用者にとっては使い勝手がいいが，習熟していない利用者にとっては結合子や反復子においての括弧の対応などが特に難しく，習熟するまではそういったエラー解決に時間を要する．そこで本論文で提案する SSFlow では，配置されたブロックを元に自動でクエリを生成したり，マウスの操作によって特定の要素を削除することで，利用者はコードを実行して結果を確認し，修正して再度実行するといった負担が低減し，SuperSQL のコード作成にかかる時間も削減されると考えられる．

3　SSFlow

SSFlow は Flutter [5] を用いて実装したデスクトップアプリケーションであり，デザインなどは FlutterFlow [6] を参考にしている．SSFlow はブロック要素をキャンバス上に追加するダイレクトマニピュレーション (DM) ツールとなっている．本論文で提案する SSFlow のメイン画面を図 1，図 2 に示す．

3.1　キャンバスエリア

図 1 の真ん中の部分をキャンバスエリアと呼ぶ．SSFlow では左側のブロック要素エリアからこの部分にブロックをドラッグ&ドロップすることで，SSFlow 内部で持っている配列に要素を追加する．こうしてブロック要素が追加されることで，後述する要素間関係表示エリアとコード生成エリアに表示するデータも更新する．

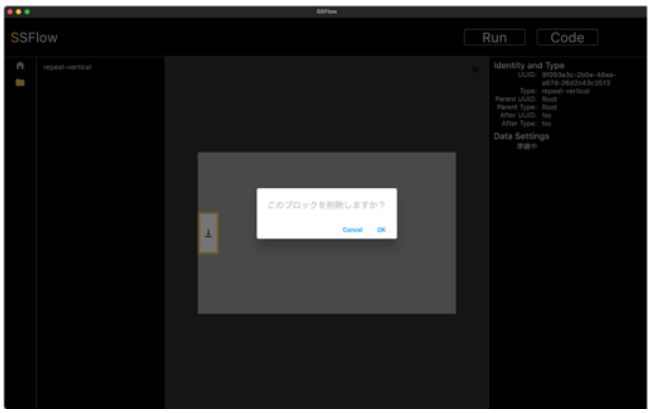

図 3　特定のブロックの削除

要素を削除したい場合には該当のブロックを右クリック (Mac では副ボタンのクリック) をすることで，その要素を削除するかの確認ダイアログが出る (図 3)．その際，その要素に子要素が含まれている場合はそれらの子要素もまとめて削除する．

3.2　ブロック要素エリア

図 1 の左側の部分をブロック要素エリアと呼ぶ．ここには現在の SSFlow で使われるブロック要素の集合がある．このブロック要素を真ん中のキャンバスエリアにドラッグ&ドロップをして操作する．ブロック要素エリアと要素間関係表示エリアは左側のアイコンボタンを押すことで切り替える．

3.3　属性エリア

図 1 の右側の部分を属性エリアと呼ぶ．ここにはキャンバスエリア上のブロックをクリックした際にその要素の情報を表示する．Text/Table Data/Function の 3 つの要素に関してはデータを保持する．そして属性エリアではそのデータの中身をテキストフィールドによって変更する．また Table Data の要素に関しては，図 4 のようにテーブル名とカラム名の 2 箇所を変更することがする．ここに記述したテーブル名が，コード生成のうちの DB パートに使われる．属性エリアとコード生成エリアはメイン画面の右上のボタンを押すことで切り替える．

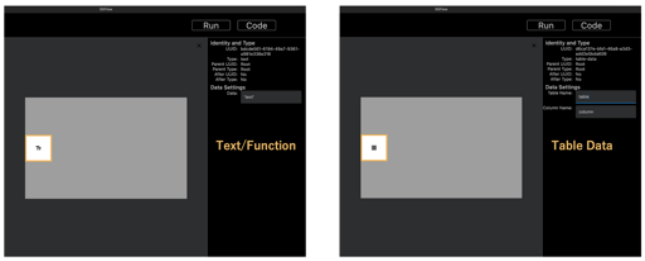

図 4　データを持つ要素の属性エリアでの表示

3.4　要素間関係表示エリア

図 2 の左側の部分を要素間関係表示エリアと呼ぶ．ここにはキャンバスエリアに配置されているブロックをもとに生成された要素間の親子関係をわかりやすくしたものを表示する．これを見ることで，どの要素がどの要素の親にあたるか，隣接する要素はどれか，などがひと目で分かるようになる．

3.5　コード生成エリア

図 2 の右側の部分をコード生成エリアと呼ぶ．これも要素間関係表示エリアと同じく，キャンバスエリアに配置されているブロックをもとに生成された SuperSQL のコードを表示する．また，右上のコピーアイコンをクリックすることでクリップボードに生成されたコードをコピーする．

4　システム設計

4.1　SuperSQL 要素の表現

本章では，SuperSQL の要素が SSFlow の実装においてどのように表されているかについて述べる．

SuperSQL の要素を layoutType として表している．layoutType の一覧は 5.1 の通りである．そしてその layoutType が Text や Table Data などデータを持っている場合には，body にそのが入ることになる．例えば，SuperSQL において'Name' というテキスト要素の場合は，SSFlow では layoutType が text であり．その body は Name ということになる．

また，それ以外のパラメーターとしては

- uuid
- parentUuid
- prevUuid

がある．これらはそれぞれ，ブロックとしての要素であり，親要素や隣接要素の uuid の情報を保持する．

5.6 にブロックの追加からコード生成までの流れを載せている通り，今回は SSElement を配列形式で追加していき，その配列をもとに親子関係を反映した木構造を構成し，要素間関係表示エリアとコード生成エリアに表示するデータを更新するという実装である．

4.2　キャンバス上のブロックの配置

本章では，キャンバス上のブロックをどのように配置するかの設計についてを述べる．キャンバスにブロックが配置されて

いる様子を図 5 に示す．図中の青枠は，水平結合子/反復子であり，その中のブロックは横方向につながっていくようになっている．同様に，垂直結合子/反復子の中にあるブロックは縦方向につながるようになる．本論文執筆時には実装は完了していないが，提案するシステムでは，ブロック要素をこのように配置することを目指した．

図 5　キャンバス上のブロックの配置

4.3　採用を見送った設計案

4.1 で述べた通り，SSFlow ではブロックの追加による SSElement 要素の変更については配列形式で実装した．本章では，SSElement 要素の変更についてのシステム設計において，採用を見送った設計案についてを述べる．その案とは，Dart 言語の Map という key/value 形式のクラスを用いて，ブロックの追加による SSElement 要素の変更について対処しようとするものである．

この案の採用を見送ったのは，木構造の方が扱いやすく，また親子関係が増えるほどネストが深くなるからである．ブロックの追加の段階では単純に配列の要素数を増やし，それぞれの parentUuid や prevUuid という親要素や隣接要素の情報を元に木構造を構成することで，その後の要素間関係表示エリアとコード生成エリアに表示するデータの更新といった処理を行う設計が可能となった．

5　SSFlow の実装

先述したように SSFlow は Flutter を用いて実装している．本章では，SSFlow のシステムの実装についての説明をする．

5.1　SSElement とブロック

まず重要なのは，SSElement と定義したものである．これは SuperSQL の略として SS を接頭語とした要素のことである．以下に SSElement の定義を示す．

```
1   // ss_element.dart
2   mixin WithUuid {
3     late final String uuid;
4     String? parentUuid;
5     String? prevUuid;
6   }
7
8   class SSElement with WithUuid {
9     final String layoutType;
10    String body;
11
12    SSElement(
13      this.layoutType, {
14      this.body = '',
15    }) {
16      uuid = const Uuid().v4();
17    }
18  }
```

そしてこの SSElement を要素として持つオブジェクトが SSFlow でブロックと呼んでいるものになる．現在の SSFlow で定義しているブロックは以下の通りである．括弧内が SSFlow における呼び名となる．

- 垂直結合子 (column)
- 水平結合子 (row)
- 垂直反復子 (repeat-vertical)
- 水平反復子 (repeat-horizontal)
- 中括弧 (block)
- 文字列 (text)
- 関数 (function)
- テーブルデータ (table-data)

これらの要素に関しては 2 章で述べた通りである．

5.2　キャンバスエリアへのブロックの追加

キャンバスエリアへのブロックの追加は，画面左側のブロックを画面真ん中のキャンバスエリアにドラッグ&ドロップすることで行う (図 6)．この操作を行うと List<SSElement>型の配列に該当ブロックが持っている SSElement 要素を追加する．図 6 のように，キャンバスへの追加の場合は親要素がないので，parentUuid は null となるが，図 7 のように親要素内にドラッグ&ドロップした場合には，その要素の uuid が新たに追加した要素の parentUuid となる．この場合は，column(3e4d3d1e...) の parentUuid が repeat-horizontal の uuid である f2cdf699... となる．こうしてできた SSElement の配列を元にして，コード生成や要素間関係の更新が行われる．

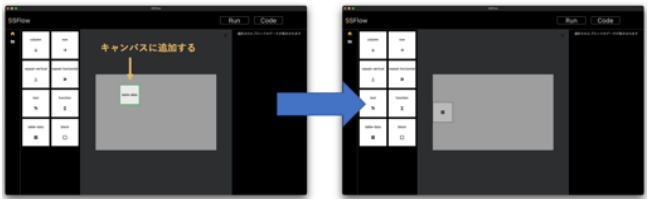

図 6　ブロックの追加方法

5.3　コード生成

5.2 で述べた通り，List<SSElement>型の SSElement の配列に変更が加わると，コード生成が行われる．コード生成は主に 3 つのパートに分けて実装しており，メディアパート，DB パート，クエリパートとしている．まずメディアパートとは

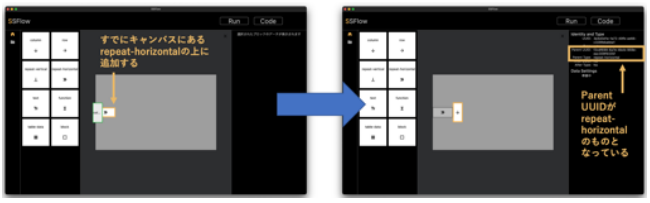

図 7　子ブロックの追加方法

アルゴリズム 1 コード生成

```
1: function generateCode(node, {separator = ""})
   ▷ node は SSElement. 最初の呼び出し時にはルート要素 (すべての親) を渡す.
2:     String result = ""
3:     if node に子要素が存在していれば then
4:         return node.data + separator
5:     for all t ← node.children do ▷ 現在ループの対象となっている node の子要素を t とする
6:         t の種類によって separator に結合子, before/after に反復子を入れる
7:         if t に子要素が存在していれば then
8:             String generated = generateCode(t, separator)
9:             generated = removeSeparator(generated) ▷ 不要な末尾の結合子を削除する
10:            result+ = before + generated + after
11:        else
12:            result+ = generateCode(t, separator)
13:    return result
```

SuperSQL における出力形式のことで，現時点の SSFlow では HTML 形式のみとしている．DB パートでは，SSElement の配列の中にある table-data 要素のうち，テーブルの情報 (図 4 の「Table Name」部分) を取得し，それらをカンマ区切りで FROM 句の後に続けて並べる．現状，WHERE 句などの結合条件などは未実装であるため，実際には結合条件などを書き加える必要が出てくる場合がある．そしてクエリパートではアルゴリズム 1 のようにして SuperSQL のコードを生成している．

このような再帰関数によってクエリパートで文字列を生成している．最後に，これら 3 つのパートを足し合わせることで SuperSQL のコードを生成している．

5.4　要素間関係の扱い

5.2 で述べた通り，List<SSElement>型の SSElement の配列に変更が加わると，要素間関係の表示を更新する．ここでは SSFlow における要素間関係がどのように扱われているかを述べる．この関係は図 2 の左側の表示エリアに表示されている．図 2 ではすべて展開されている状態であるが，右側のアイコンボタンを押すことでその要素の子要素の情報を閉じる．Flutter ではオブジェクトの表示に Widget というものが使われており，ここではその中の ExpansionTile という開閉可能な Widget を使用している．子要素がない SSElement に関しては，開閉しない ListTile という Widget を使用しており，要素間関係を扱う形式として List<Widget>型の Widget の配列を生成している．実装はアルゴリズム 2 のようになる．

アルゴリズム 2 要素間関係の扱い

```
1: function buildExpansionTile(node, {offset = ""})
   ▷ node は SSElement. 最初の呼び出し時にはルート要素 (すべての親) を渡す.
2:     List<Widget> result = []
3:     for all t ← node.children do ▷ 現在ループの対象となっている node の子要素を t とする
4:         if t に子要素が存在していれば then
5:             Widget tile = ExpansionTile(
                   title : Text(offset + t.data),
                   children : buildExpansionTile(
                       t, offset : '  ' + offset,
                   ),
               ) ▷ buildExpansionTile() を呼び出す際に，offset には空白を追加する
6:         else
7:             Widget tile = ListTile(
                   title : Text(offset + t.data),
               )
8:         result.add(tile)
9:     return result
```

図 8 の例では，column と repeat-vertical のような親子関係にはオフセットが付与されており，row の子要素のうち兄弟関係にある 2 つの table-data 同士ではオフセットがないことがわかる．

図 8　親子関係の要素にはオフセットを付与

5.5　Undo と Redo

SSFlow では画面上の矢印を操作することで操作を取り消したり，やり直したりする．操作の様子を図 9 に示す．ここではその操作を取り消す Undo と、やり直す Redo についての実装について述べる．これまでに述べた通り，SSFlow では SSElement の配列を元にコード生成などが行われる．そこで，Undo と Redo を実現するために 2 つのスタックを用意して，SSElement の配列をプッシュとポップにより保存や取り出しを行う．

図 9　Undo と Redo

5.6　SSFlow の実装のまとめ

ここまでの流れをまとめたものが図 10 となる．まずキャンバスにブロックを追加することによって SSElement の配列を生成する．その配列をもとに要素間関係表示エリアとコード生成エリアに表示するデータを更新する．

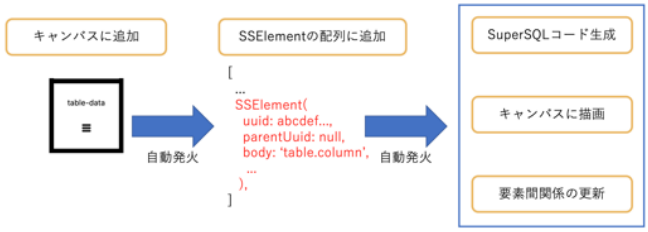

図 10　ブロックの追加からコード生成までの流れ

6　評　　価

本章では評価として，本論文で提案している SSFlow を類似ツールとの機能の比較による評価と，実際に授業で作成したクエリを SSFlow でどこまで再現可能かという再現度及び所要時間の比較による評価で，提案手法の有用性について考察する．

6.1　類似ツールとの比較

SSFlow の No-Code ツールとしての性能を測るために，世の中ですでに使われている FlutterFlow [6] と Bubble [7] という 2 つの No-Code ツールとの機能比較を行った．これらを比較対象として選択した理由としては，どちらもすでに多くの人に使われている No-Code ツールであるからである．また FlutterFlow は SSFlow の実装において参考にしている部分が多い．本論文で提案した SSFlow と，FlutterFlow 及び Bubble との No-Code ツールにおける機能比較を表 3 に示す．

表 3 より，本論文で提案した FlutterFlow は No-Code ツールとして使われるには，エラー表示やプレビュー表示などの基本機能が足りていないことが分かる．現在は未実装ではあるが 8.2 で述べるように，ファイル読み込み機能をつけてすでに作成しているコードを読み込んで，それをキャンバスに反映させるという機能をつけることで，他のツールにはない特徴となる．またそれはクエリからレイアウトが判別可能であるという SuperSQL の特徴も活かされる．他の 2 つのツールにある機能のうち，SSFlow にはないものでバージョン管理の機能がある．これはその時のキャンバスの状態に名前をつけることで，複数のバージョンを管理するものである．No-Code ツールでは複数のバージョンを同時に管理しておくことが重要であるので [8]，この機能を SSFlow に組み込むことで，習熟度に関係なく使いやすいツールになっていくと考えられる．以上より，現在の実装状況では本論文で提案した SSFlow は，他のすでに世の中で使われている No-Code ツールと比較すると必須機能が不足しているなどの課題があるが，その部分を実装し，すでに作成しているコードの書かれたファイルを読み込み，キャンバスに反映させるという機能を実装することで，他のツールにはないインタラクティブを達成することが可能になるため，その分野において有用になると考えられる．

表 3　類似ツールとの機能比較

クエリ作成支援機能	FlutterFlow	Bubble	SSFlow
ファイル読み込み	×	×	×
Undo（取り消し）	○	○	○
Redo（やり直し）	○	○	○
コード自動生成	○	○	○
ブロックの追加・削除	○	○	○
ブロック間関係の表示	○	×	○
要素の属性表示	○	○	○
要素の属性編集	○	○	○
プレビュー表示	○	○	×
エラー表示	○	○	×
バージョン管理	○	○	×

6.2　クエリの再現度と所要時間

表 4　SuperSQL 表現の再現度

クエリ作成支援機能	SSFlow
FOREACH 句の生成	×
GENERATE 句の生成	○
FROM 句の生成	△
レイアウト構造の生成	○
各関数の追加	○
テーブルデータの追加	○
文字列の追加	○
先頭の asc や null	×
要素のデータ編集	○
装飾子の編集	×
関数の入れ子	×

ここでは，過去に授業で作成した SuperSQL クエリの再現度と所要時間について，SSFlow と SSedit を組み合わせた方法と SSedit のみで作成した場合で比較する．クエリの再現をする際に使用したクエリ例および，このクエリの再現で，SSFlow と SSedit の切り替えのタイミング，すなわち SSFlow を使用して生成したクエリを以下に示す．これ以降のクエリの編集は SSedit で行った．再現度については，各表現について再現可能かそうでないかの表を作成して比較することとし，結果を表 4 に示した．これをみると，基本的なレイアウト構造の生成や GENERATE 句などの生成はできていることがわかる．FROM 句については，2 つ以上のテーブルになったときにそれ

らの結合条件などを反映させる機能がないため，その部分はその後の SSedit でクエリを修正する必要がある．よく使われる表現のうち，装飾子の反映は No-Code の方が扱いやすくなるので，実装する必要がある．

次にクエリ作成の所要時間を比較する．SSFlow と SSedit を組み合わせた方法では，SSFlow で 142 秒，その後引き継いで SSedit で 243 秒で合計 385 秒を要した．SSedit のみでクエリを作成した場合には 419 秒を要した．これらの数値を比較すると，全体の時間を 8.1%削減した．また SSedit を操作していた時間で比較すると，42%削減した．

SSedit のようなテキストベースのクエリ作成ツールの時間を削減することは，特に SuperSQL のクエリ作成に習熟していない利用者にとっては，ブロック要素型 No-Code のように単純な操作でクエリ作成を行うこととなり，有用である．

```
1  -- 評価に使用したクエリ例
2  -- advertisement_index.ssql
3  GENERATE HTML
4  {
5    {
6      {a(' トップ', 'index.html')}!
7      ' 広告リスト'@{font-size=24, class='bg-
           primary text-white font-weight-bold'}!
8      {'ID'@{width=100}, ' メディア分類
           '@{width=300}, ' タイトル
           '@{width=300}, ' 説明
           '@{width=500}}@{class='thead-light'}!
9      [(asc)a.id@{width=100}, m.name@{width
           =300}, link(a.title@{width=300}, '
           advertisement_show.ssql', a.id), a.
           description@{width=500}]!10%
10   }@{cssfile='jscss/bootstrap.min.css, jscss/
         advertisement_index.css'}
11 }@{title=' 広告代理店| 広告リスト
       ' , table-align='center', width=1200,
       class='table table-sm'}
12 FROM advertisement a inner join media m on a.
       media_id = m.id;
```

```
1  -- SSFlow での生成結果
2  -- advertisement_index_ssflow.ssql
3  GENERATE HTML
4  {
5    { a(' トップ', 'index.html')} !
6    ' 広告リスト' !
7    { 'ID' , ' メディア分類' , ' タイトル
         ' , ' 説明' }
8    [ a.id , m.name , link(a.title, '
         advertisement_show.ssql', a.id) , a.
         description ]!
9    }
10 }
11 FROM a, m
```

7 関連研究

本論文ではブロック要素を用いた No-Code ツールを提案しているが，近年，No-Code/Low-Code は非常に大きなトレンドである [9]．COVID-19 の流行によりオンラインでの学習が行われるようになってきた現在ではさらにその兆候が見える．No-Code，特にブロック型のプログラミングツールは Scratch [10] や Blockly [11] などすでに教育の分野で使われているものも多い．ブロック型のツールではオブジェクトを接続する際のバリデーションの適用など，インタラクティブなフィードバックがあることが特徴である．そのため，比較的経験の浅い学習者や若い学習者が使用の対象となることが多く，教育にこういったツールを用いる影響や貢献度の調査も頻繁に行われている [12] [13] [14] [15]．ブロック型のプログラミングツールを用いた教育は，パズルで遊んでいるような感覚だが，単なるゲームとは異なり，知識とベストプラクティスを共有する，非常に効果的なフレームワークとなっている [16]．さらに，このような直接操作 (DM) のヴィジュアルプログラミングツールが教育に取り込まれている現状から，視覚障害者を対象とした，タッチスクリーンの iPad アプリケーションである Blocks4All のようなツールも開発されている [17]．また，教育の分野以外ではロボットプログラミングなどのスマート環境に関連する分野でのツールの開発も行われている [18] [19] [20] [21]．このように No-Code ツールのトレンドは加速しており，今後の発展に向けての基礎や研究の道筋の提案もされている [22]．本論文で提案する SSFlow もこのトレンドを踏襲し，経験の浅い利用者の負担を減らし，開発効率の向上を促進することを目的としている．

8 おわりに

8.1 結　　論

本論文では，SuperSQL のクエリ作成に習熟していない利用者のクエリ作成の負担を軽減するために，ブロック要素をドラッグ&ドロップで操作することでキャンバスに追加し，またそのレイアウトを確認することで，SuperSQL での開発を容易にし，クエリの修正回数を減らすことを目的とした，No-Code ツール (SSFlow) を提案し，実装した．SSFlow は配置されたブロックを元に自動でコードを生成するため，SuperSQL 利用者の負担となる括弧の対応を考える必要がなくなる．また，マウスの操作によって特定の要素を削除するため，クエリの修正にかかる負担を減らすことが可能になっている．これにより，複雑なクエリも効率よく編集可能である．また SSFlow を用いることで，SSedit の操作時間を 42%削減したことにより，SuperSQL のクエリ作成に習熟していない利用者がテキストベースのツールを使うことによる負担を低減した．そして，SuperSQL のクエリ作成にかかる時間も 8.1%削減した．

8.2 課題・展望

6 章で述べたように本論文で提案した現在の SSFlow の機能のみでは，SSedit の機能を網羅できていないために SuperSQL クエリ作成ツールの 1 番目の選択肢とはならない．そのため，まずは SSedit にある機能を SSFlow にも追加することが課題

となる．そのうち DM ツールにおいて特に重要な複数キャンバス，編集したブロックをまとめてコピー&ペーストするような機能 [8] を中心に実装する．

以下に具体的な実装予定を，必須機能やその他の機能などでいくつかの段階に分けて示す．

（1） ブロックのキャンバスへの配置 ※段階 1

（2） 外部ファイルの読み込み ※段階 2

（3） 生成されたコードをローカルファイルとして保存 ※段階 2

（4） SuperSQL を埋め込んで直接クエリを実行 ※段階 2

（5） 生成されたコードを直接編集 ※段階 3

（6） ショートカットキーの追加 ※段階 3

これらの機能実装後は，この SSFlow が利用者の習熟度に関係なく使いやすいツールとなるように，利用者のフィードバックを受けながら機能追加や改善を行う予定である．これらの機能を実装するまでの SuperSQL のクエリ生成には，クエリ生成の初期段階として，本論文で提案した SSFlow を用い，それ以降の修飾子の調整やクエリ実行に SSedit を用いる．

文　献

[1] Toyama lab., “SuperSQL”, http://ssql.db.ics.keio.ac.jp.

[2] Motomichi Toyama, “SuperSQL: An Extended SQL for Database Publishing and Presentation”, Laura M. Haas, Ashutosh Tiwary, editors, Proceedings ACM SIGMOD International Conference on Management of Data, pp. 584–586, 1998.

[3] 春野健吾, 五嶋研人, 遠山元道, “直接操作による SuperSQL クエリ作成支援”, データ工学と情報マネジメントに関するフォーラム (DEIM), 2015.

[4] 木谷将人, 五嶋研人, 遠山元道, “SuperSQL クエリ作成支援システムの開発”, データ工学と情報マネジメントに関するフォーラム (DEIM), 2014.

[5] Google, “Flutter”, https://flutter.dev.

[6] “FlutterFlow”, https://flutterflow.io.

[7] “Bubble”, https://bubble.io.

[8] Dor Ma’ayan, Wode Ni, Katherine Ye, Chinmay Kulkarni, Joshua Sunshine, “How Domain Experts Create Conceptual Diagrams and Implications for Tool Design”, Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems, pp. 1–14, 2020.

[9] Forbes, “The Most Disruptive Trend Of 2021: No Code / Low Code”.

[10] J. Maloney, L. Burd, Y. Kafai, N. Rusk, B. Silverman, M. Resnick, “Scratch: a sneak preview [education]”, Proceedings. Second international conference on creating, connecting and collaborating through computing, 2004., pp. 104–109, 2004.

[11] Google, “Blockly”, https://developers.google.com/blockly/.

[12] Aneliya Ivanova, “A Concept of Visual Programming Tool for Learning VHDL”, IOP Conference Series: Materials Science and Engineering, Vol. 1031, No. 1, p. 012120, 2021.

[13] Brian Broll, Akos Lédeczi, Peter Volgyesi, Janos Sallai, Miklos Maroti, Alexia Carrillo, Stephanie L. Weeden-Wright, Chris Vanags, Joshua D. Swartz, Melvin Lu, “A Visual Programming Environment for Learning Distributed Programming”, Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education, SIGCSE ’17, pp. 81–86, 2017.

[14] Mazyar Seraj, Eva-Sophie Katterfeldt, Kerstin Bub, Serge Autexier, Rolf Drechsler, “Scratch and Google Blockly: How Girls’ Programming Skills and Attitudes Are Influenced”, Proceedings of the 19th Koli Calling International Conference on Computing Education Research, Koli Calling ’19, 2019.

[15] Mazyar Seraj, “Impacts of Block-Based Programming on Young Learners’ Programming Skills and Attitudes in the Context of Smart Environments”, Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE ’20, pp. 569–570, 2020.

[16] Luis Corral, Ilenia Fronza, Claus Pahl, “Block-based programming enabling students to gain and transfer knowledge with a no-code approach”, Proceedings of the 22st annual conference on information technology education, SIGITE ’21, pp. 55–56, 2021.

[17] Lauren R. Milne, Richard E. Ladner, “Blocks4All: Overcoming Accessibility Barriers to Blocks Programming for Children with Visual Impairments”, Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, pp. 1–10, 2018.

[18] Gaoping Huang, Pawan S. Rao, Meng-Han Wu, Xun Qian, Shimon Y. Nof, Karthik Ramani, Alexander J. Quinn, “Vipo: Spatial-Visual Programming with Functions for Robot-IoT Workflows”, Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems, pp. 1–13, 2020.

[19] Enrique Coronado, Dominique Deuff, Pamela Carreno-Medrano, Leimin Tian, Dana Kulić, Shanti Sumartojo, Fulvio Mastrogiovanni, Gentiane Venture, “Towards a Modular and Distributed End-User Development Framework for Human-Robot Interaction”, IEEE Access, Vol. 9, pp. 12675–12692, 2021.

[20] Alexandre G. De Siqueira, Pedro G. Feijóo-García, Sean P. Stanley, “BlockXR: A Novel Tangible Block-Based Programming Platform”, 2021 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), pp. 1–4, 2021.

[21] Mazyar Seraj, Serge Autexier, Jan Janssen, “BEESM, a Block-Based Educational Programming Tool for End Users”, Proceedings of the 10th Nordic Conference on Human-Computer Interaction, NordiCHI ’18, pp. 886–891, 2018.

[22] Ahmed ElBatanony, Giancarlo Succi, “Towards the no-code era: A vision and plan for the future of software development”, Proceedings of the 1st ACM SIGPLAN international workshop on beyond code: No code, BCNC 2021, pp. 29–35, 2021.