

GUIを用いた SuperSQL クエリ生成支援ツール

前田 玲緒[†] 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: [†]{maeda,goto}@db.ics.keio.ac.jp, ^{††}toyama@ics.keio.ac.jp

あらまし Web ページを生成する際にデータベースを参照し、表のような形式でデータを提示するようなことはよく求められる。しかしながら実際に格納されているデータを取得するにはデータベースに関する知識、構造化された web ページの生成には php や html の知識が必要であり、一つのページを生成するだけでも多くの手間がかかってしまう。そこで本研究では Swing を用いることで視覚的に理解しながら SQL の拡張言語である SuperSQL を扱えるようにする。ユーザはデータベースからどのデータを抽出し、どのようにレイアウトするかをグリッド形式の作業画面に属性オブジェクトを配置し、グルーピングなどの修飾をすることで指定する。そのコンポーネントから構造情報を解析し、レイアウトに応じた web ページを生成する SuperSQL クエリを生成する支援システムを提案する。

キーワード データベース, SQL, SuperSQL, Swing, 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。しかし、求めている出力結果を得るためのクエリを書くためには SuperSQL の文法を理解しクエリから得られる出力結果をイメージする必要がある。SuperSQL を扱いやすくするための様々なシステムは提案されてきたが、それらは基本となるクエリ自体は自分で記述する必要があったりレイアウト構造の制限を考慮する必要があったりと事前知識が必要になるようなことが多かった。[5] [6]

本研究ではユーザが SuperSQL に関する構文知識がなくても扱えるような SuperSQL のクエリ生成を支援するツールを提案する。グリッド画面上にユーザは自由にレイアウトを指定し、システム側がレイアウトを解析しクエリの生成を行なうためユーザは直感的な操作だけで SuperSQL を利用する事が出来る。

2 SuperSQL

この章では本論文で改善を試みる SuperSQL について簡単に述べる。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 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

結合子は取得したデータをどの方向に結合するかを指定する

演算子である。

- 水平結合子 (,)

データを横に結合して出力

例: Name, Salary

Name	Salary
------	--------

- 垂直結合子 (!)

データを縦に結合して出力

例: Name! Salary

Name
Salary

2.2 反復子

反復子は取得したデータを指定された方向に結合し続ける演算子である。

- 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方向に繰り返し表示する。

例: [Name],

Name1	Name2	...	NameN
-------	-------	-----	-------

- 垂直結合子 ([]!)

データインスタンスがある限り、その属性のデータを縦方向に繰り返し表示する。

例: [Name]!

Name1
Name2
...
NameN

また、反復子は構造を指定するだけでなく属性同士の関連をネストによって表現できる。

[プロジェクト名]!, [従業員名]!, [給与]!

と記述した場合、この属性間に関連はなくただそれぞれの一覧が出力されるだけであるが、

[プロジェクト名, [従業員名, 給与]]!

と記述した場合には各プロジェクトごとに従業員と給与の一覧が出力されるようになる。

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、文字のサイズ、位置、横幅、縦幅などの情報を付加できる。これらを指定するには属性名と装飾演算子 (@) を用いて指定し以下のような形式で記述する。

<属性名>@{<装飾指定>}

装飾指定は”装飾子の名称 =その内容”で指定し、”, ”を用いて複数個の装飾を同時に行うことができる。

2.4 関数

SuperSQL ではいくつかの関数が用意されており、ここでは image 関数について説明する。image 関数を用いると画像を表示することが可能となる。引数には属性名、画像ファイルの存在するディレクトリのパスを指定する。

image(id, '/pic')

3 提案システム

3.1 システム概要

本研究ではユーザが手軽に利用できる Java Swing [3] によって実装を行なった。ユーザは SuperSQL クエリ自体を記述する必要はなく、GUI システム上でどういったレイアウトにしたいかを実際に値を配置することで指定し、その構造を解析し同様のレイアウトを出力結果とする SuperSQL クエリをシステムが生成する。また、ユーザがデータベースに詳しくなくても使えるように接続するデータベースを指定すればテーブルの一覧を取得し、必要なテーブルがあればそのテーブルの情報を取得する。接続するデータベースは PostgreSQL [4] に限定している。レイアウトを指定する場合にも取得したテーブル情報から選択すればいいため、スペルミスや構文エラーを意識しないでもいいようになっている。

3.2 データベース画面

id	name	salary	mana...	byear	syear	pict	d_id
97	山口 淳	7000	110	1990	2011	97.png	10
423	山根 ...	12000	47	1972	1994	423...	19
147	会津 ...	19000	199	1972	1994	147....	20
210	岩井 ...	17000	199	1960	1983	210....	14
1110	江藤 ...	6000	33	1967	1988	111....	43
35	遠藤 ...	5000	32	1967	1989	35.png	65
157	山田 ...	12000	199	1955	1975	157....	35
129	小川 ...	10000	199	1956	1977	129....	10

図 1 データベース画面

データベース画面では接続するデータベースと取得するテーブルの指定、実際に表示する値の指定を行う。データベースとの接続ではユーザは接続するデータベースとユーザ名を指定する。データベースとの接続が成功すると存在するテーブルの名前の一覧を取得し、さらに出力したい情報があるテーブルを選択することでそのテーブルの情報を取得する。ユーザはテーブル形式で表示されている情報からどのようなデータがあるかを確認しながら出力するデータを選ぶことができる。また、表示されているセルの配置を変えたくなった場合にはドラッグアンドドロップでセル同士を入れ替えることができる。

3.3 メイン画面

図 2 メイン画面

メイン画面では実際にデータをどのように表示したいかをグリッド形式で示すための画面となっている。ここでは先のデータベース画面から選択したデータをどこに配置するかを指定したり、文字列を追加したりセル自体の修飾情報を指定する作業を行う。以下ではそれぞれの機能について説明していく。

3.4 セルのサイズ

3.4.1 セルの横幅

グリッド形式の各セルには初期サイズが設定されているが、個別にセルのサイズを変更できる。セルを選択した状態で横拡大をすると初期のセル1つ分のサイズが指定されたセルに加算される。また、細かく数値を指定したい場合にはセルを選択した状態でメイン画面のセルの取得を行なった後に、セルサイズを数値で指定することができる。セルの大きさが変更されたらメイン画面のレイアウト表示も指定された値を反映し更新される。

3.4.2 セルの縦幅

縦幅を加算したい場合にはメイン画面の縦拡大を押すことでセルの縦幅が1マス分が加算される。縦幅が2以上のセルがある場合その左右のセルは入れ子の表現の扱いになる。

3.4.3 セル同士の結合

関連性が高いセル同士を結合することで一つのセルとして扱うことができる。メイン画面のセルを選択した状態で右と結合、または下と結合を押すと指定されたセル同士が結合される。1つのセルとして扱うことで後述の反復子の指定をする場合やサイズの指定をする場合にセル全体を指定することができるようになる。また、結合した順序によって構造の違いを表現するこ

とができる。例えば、以下の図3のようなレイアウトを作りたいとする。

山田 次郎			
慶應義塾	入学	2014年	4月
	学部	理工学部	
	学科	情報工学科	

図3 目標とする出力結果

まず単純に必要なデータを選択しレイアウトと同じような位置に指定してクエリを生成したものとそのクエリを SuperSQL で実行した時の出力結果を図4に示す。

レイアウト画面

☐ 山田 次郎	☐	☐	☐	☐
☐ 慶應義塾	☐ 入学	☐ 2014年	☐ 4月	☐
☐	☐ 学部	☐ 理工学部	☐	☐
☐	☐ 学科	☒ 情報工学科	☐	☐

クエリ実行

生成されたページ

山田 次郎			
慶應義塾	入学	2014年	4月
学部	理工学部		
学科	情報工学科		

図4 単純に値を配置した時のクエリの生成

単純に配置しただけではこのようにイメージとは異なる出力結果になってしまう。そこでまずは図3のレイアウトに近づけるために慶應義塾のセルの縦幅を3行分に拡大する。こうすることで図5のように入れ子のような表現が可能になる。

レイアウト画面

☐ 山田 次郎	☐	☐	☐	☐
☒ 慶應義塾	☐ 入学	☐ 2014年	☐ 4月	☐
☐	☐ 学部	☐ 理工学部	☐	☐
☐	☐ 学科	☐ 情報工学科	☐	☐

クエリ実行

生成されたページ

山田 次郎			
慶應義塾	入学	2014年	4月
	学部	理工学部	
	学科	情報工学科	

図5 ネストを考慮したクエリの生成

次に 2014 年のセルを選択した状態でメイン画面の右と結合を押すことで、2014 年と 4 月のセルが結合され 1 つのセルとして扱えるようになる。図5の出力結果では入学の行がセル3つ分になって表示が整っていなかったのが、先の結合したセルのサイズを変更することで表示を整えていく。結合したセルを選択した状態でメイン画面のセルの取得を押すと、対象のセルの隣にあるドロップダウンリストに選択されているセルの内部セル一覧が図6のように表示される。

個別のセルのサイズ指定も可能であるが、今回は全体のサイズを整えるためにドロップダウンリストからセル全体を指定し、セルのサイズを図7のように変更する。セルのサイズを変更す

2014年
4月

対象のセル：
✓ セル全体

セルの取得

セルサイズ：
320

更新

図6 セル情報

るとレイアウトの表示も自動で切り替わるため、他のセルとのサイズがあっているかの確認もしやすくなっている。

セルサイズ変更前

☒ 2014年	☐ 4月
☐ 理工学部	☐

セルサイズ変更後

☒ 2014年	☐ 4月
☐ 理工学部	☐

セルサイズの変更

対象のセル：
セル全体

セルの取得

セルサイズ：
160

更新

320→160

図7 セルサイズの変更

このレイアウトからクエリを生成し実行すると図3と同じレイアウトを持つ出力結果を得ることができる。

レイアウト画面

☐ 山田 次郎	☐	☐	☐	☐
☒ 慶應義塾	☐ 入学	☒ 2014年	☐ 4月	☐
☐	☐ 学部	☐ 理工学部	☐	☐
☐	☐ 学科	☐ 情報工学科	☐	☐

クエリ実行

生成されたページ

山田 次郎			
慶應義塾	入学	2014年	4月
	学部	理工学部	
	学科	情報工学科	

図8 セルのサイズを考慮したクエリの生成

3.5 関数の指定

メイン画面でセルを取得し、個別のセルを選択している状態でのみ各関数の追加を行うことができる関数の追加が可能になる。image 関数ではパスの指定が必要なため関数の追加画面では GUI によってどのディレクトリを参照するかを選択することでパスを取得する。パスを取得した後はそのパスが正しいかを確認した上で関数の追加画面のボタンを押すことでウィンドウを閉じることができる。内部ではパスを取得すると属性名を image 関数を適用した形に上書きする。

3.6 反復子の追加

反復子を追加したい場合にはセルを選択した状態で縦反復の追加、または横反復の追加を押すとセルの色が変化し反復子が適用されたことを示す。2.2 節で述べたように反復子を使う場合にはネストを考慮する必要がある、この差異を表現するために 1 つのセルの全体に対して反復子を追加する。そのため、複数

例として各店舗と商品の一覧を表示する場合を考える。レイアウト上で図9のように全体にのみ反復子を適用した場合には店舗名や階層などが同じでも並んで何回も表示されてしまう。

図 9 全体にのみ反復子を適用した場合

図 10 全体と店舗情報に対し反復子を適用した場合

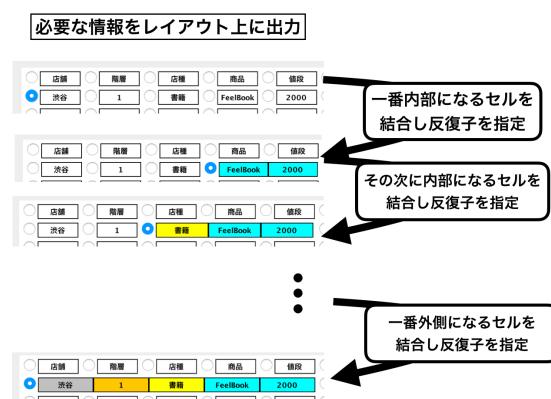


図 11 入れ子を考慮して反復子を指定する方法

図 12 入れ子を考慮した場合の出力結果

4.1 レイアウトの可否

(1) 線によって分割された全ての図形が長方形である

- (1) 線によって分割された全ての図形が長方形である
- (2) 各長方形の中に余分な線分が存在しない
- (3) 表構造全体が長方形である

しかしながらユーザにとってこれらの判断や構造の設計は手間がかかる上難しく、本システムでは似たような構造を持つ実行可能なクエリを生成することで正しくない表構造でも SuperSQL を実行できるようにしている。図 8 では本来表構造として山田次郎のセルの横幅があっていないため正しくないが、似たような表構造になるようにクエリを生成する。

グリッド形式のレイアウトから配置されている情報を把握するために各セルは自身の対応する以下の情報を有している。

- 座標
- 属性名
- 装飾子

関数

以降では説明のために図 13 のような記法を用いる。

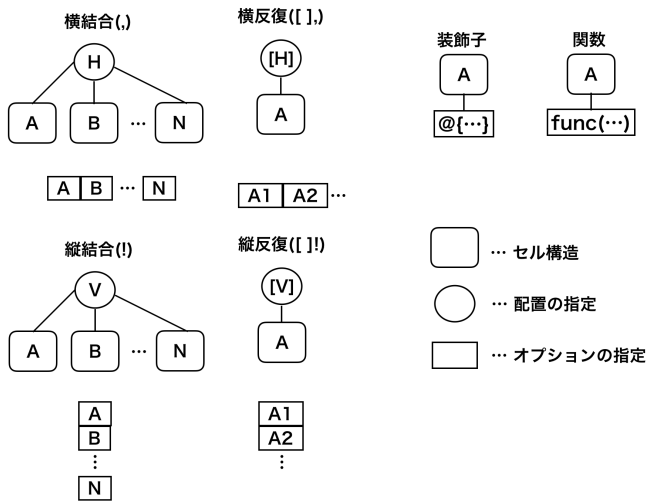


図 13 構造の記法

セルの移動や結合があるたびにその情報をレイアウト表示とともに更新していく。レイアウトの指定の際にセル同士を結合している場合には結合した段階で縦結合や横結合を用いて属性を結合するため、縦結合を用いてから横結合をする場合と横結合を用いてから縦結合をする場合には出力結果が異なり、最初に結合した属性ほど関連性が高いと判断し、入れ子の優先度を決定する。結合順序による構造の違いを図 14 に示す。

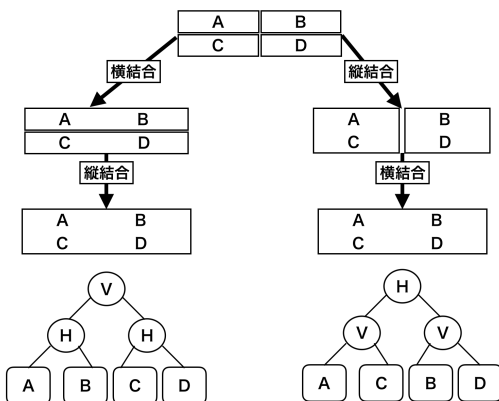


図 14 結合順序による構造の違い

最終的に生成される形が同じだとしても、結合の順序によってその内部構造は異なってくる。ユーザが結合した順番が関連性の高い順番である前提で解釈しているため、ユーザは関連性が高い順にセルの結合を行っていくことで入れ子の構造を生成できる。

入れ子の構造解析

本システムでの入れ子構造の認識は行数で判断しており、複数行にまたがるセルがあると入れ子構造としてセル同士を構造化し構造を更新する。例として図 15 ではまず横結合を行なったセルに対して入れ子構造になるようなセルが配置された場合の構造の変化を示している。

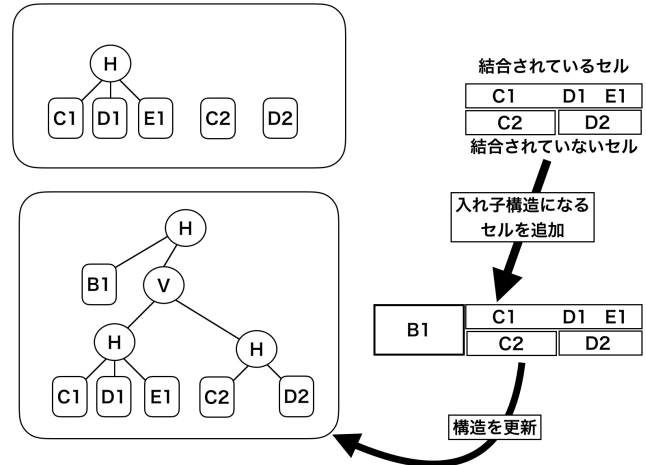


図 15 入れ子による構造の変化

入れ子構造になったセル集合のうち入れ子内部の単一行のセルを全て同行で横結合したものととして構造化して扱う。そのため、結合されていなかったセルも入れ子内部で結合されたものと同じように構造化される。入れ子が多重になっても同様に構造の更新が行われる。図 16 の例だと外側の入れ子構造を生成し、その内部に図で生成された入れ子構造が組み込まれる構造になっている。

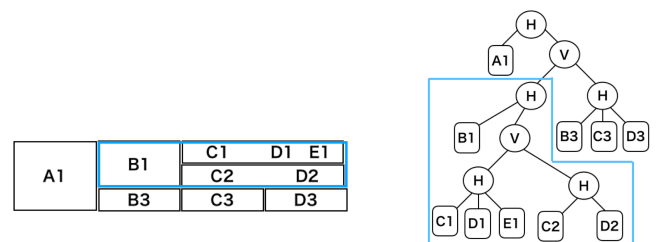


図 16 多重の入れ子

レイアウト全体の構造解析

クエリを生成するためにはレイアウトがどのように指定されているかを理解し、同じ構造になるようなクエリにどうやって対応させていくかを考える必要がある。セル単位の構造はレイアウト上で決定しているため、全体の解析では基本的に構造化された行ごとに構造解析を進めていく。途中で入れ子構造が存在するレイアウトの場合には前述のように入れ子構造の行として構造が決定しているため、次の構造化された行と縦結合を行う。単一の行に取まっている入れ子構造でないセル同士は横結

合で結合することで構造化し、構造化された行を縦結合を用いて次の行と結合する。以降の行も同様にして解析を進めていくことでセル単位の構造からレイアウト全体の構造が決定する。一例を図 17 に示す。

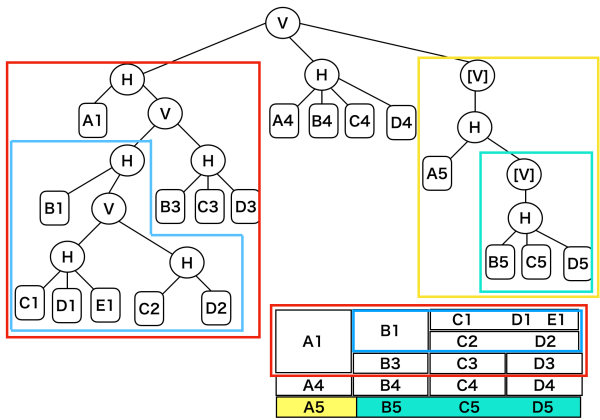


図 17 全体の構造解析

構造関係を理解することでクエリの生成も可能になる。解析された構造に対し深さ優先探索をも用いて各セルに保存されている属性名と装飾子、関数などを取得し SuperSQL のクエリとして適切なクエリを生成する。

5 実 験

提案インターフェースを評価するため、評価実験を行う。本実験ではクエリの直接記述の場合と本システムを用いた場合の比較を行うために、SuperSQL クエリを直接記述でも生成できる被験者 10 人に対してそれぞれの方法でクエリを生成してもらいクエリ生成にかかった時間と最終的なクエリができるまでのコンパイル回数を比較した。図 18 にクエリの生成にかかった時間を示す。

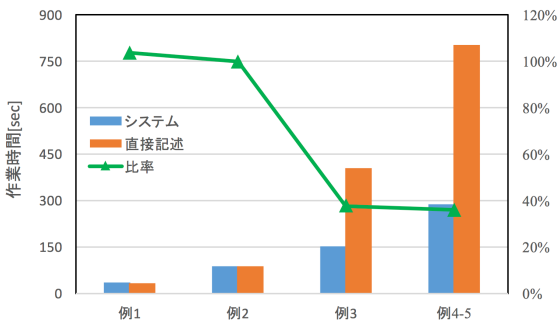


図 18 クエリ生成時間の比較

例 1 や例 2 ではクエリの生成時間にほとんど差がないが、例 3 や例 4-5 では本システムでの作業時間が直接入力の場合に比べ 40% ほどの時間でクエリの生成を行っている。例 1 や例 2 ではクエリが単純であり、クエリから結果もイメージしやすいため直接入力でもエラーやレイアウト悩まずにクエリを生成

できたためだと考えられる。一方で入れ子構造が複雑な場合には直接入力ではイメージが掴みにくく、関数を指定する場合にはその構文でエラーが起きてしまうことが多かった。本システムでは関数を指定する場合には前述のように GUI でパスの指定などを行うだけなので、構文に関するエラーは起きなかったためその分クエリの生成時間を短縮できたと考えられる。次に実際にコンパイル回数に変化があったかを確認するために、コンパイル回数を図 19 に示す。

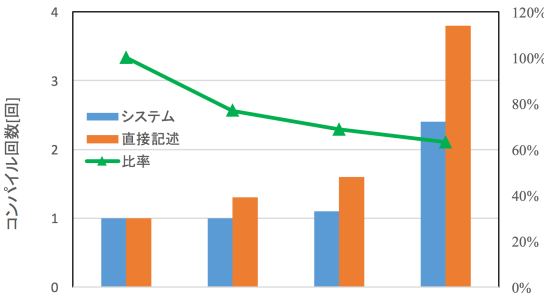


図 19 コンパイル回数の比較

実際にコンパイル回数も減らせていることがわかる。直接入力の際にコンパイル回数が増える原因としてエラーの発生や途中経過の確認などのために一旦クエリを実行しレイアウトの確認を行うことなどが挙げられる。本システムではこのような確認をする必要がないため、コンパイル回数を減らすことができている。理想的にはコンパイル回数は生成するファイル数であるが、例 3 や例 4 では本システムでもそれよりは少し多くなってしまう。この原因は where 句を自分で書く必要があり、システムが自動的に生成したエイリアスとユーザの認識が異なっていたり、単純に where 句を書き間違えていたりであったため今後改善すべき点である。

6 ま と め

6.1 結 論

本研究では GUI 上でレイアウトを指定することで SuperSQL クエリの生成を支援するシステムを開発した。GUI で操作することで文法やスペルミスに気にならず、SuperSQL の初心者や構文知識のないユーザでもクエリを生成できるようになった。

6.2 今後の課題

現在の本システムは SuperSQL の全ての機能を扱っているわけではなく、装飾子や関数などの一部にしか対応していない。SuperSQL で表現できることは本システムにも取り入れていくべきであり、今後の課題となる。

また、本システムを用いて生成したクエリを後々変更したくなった場合、クエリからレイアウト構造の推定はできないため、もう一度ははじめから GUI 上でレイアウトを指定する必要がある。今後は双方向的な対応をできるシステムを検討する。

文 献

- [1] SuperSQL: <http://ssql.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] Java Swing: <https://docs.oracle.com/javase/jp/9/docs/api/javax/swing/package-summary.html>
- [4] PostgreSQL, <http://www.postgresql.org/>
- [5] Kengo Haruno, Kento Goto, Motomichi Toyama. Modern Web Page Generation by Using Exclusively SuperSQL. In Proceedings of the 20th International Database Engineering & Applications Symposium, pp.340-345, 2016.
- [6] 吉沢 拓, 遠山 元道. ”視覚的インターフェースによる SuperSQL クエリ生成”, 慶應義塾大学, 2008.