SStest: SuperSQL 開発者のためのテスト支援ツールの開発

中井梨真子† 五嶋 研人† 木谷 将人† 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: †{rima,goto,masato}@db.ics.keio.ac.jp, ††toyama@ics.keio.ac.jp

あらまし 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 の拡張言語である. ユーザーに品質の高い SuperSQL を提供するために, テストを行うことは非常に重要である. 本論文では, SuperSQL の開発を容易にし, 品質を向上させるために, テストツール (SStest) の提案と実装を行った. SStest では GUI 上から SuperSQL のテストケースの管理を行うことができ, 複数出力される SuperSQL の生成物すべてにおいて検査を行うことができる. GUI 上から操作が行えるため, SuperSQL の開発者が簡単にテストの実行・追加・削除を行うことができる.

キーワード SQL, SuperSQL, RegressionTest

1. はじめに

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 の拡張言語である. 通常の SQL では, シンプルでフラットな表しか再現することができないが, 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 を用いることで, HTML と PHP を用いた Web ページを一般的な方法に比べて, はるかに少ない行数で生成することができる.

しかしながら, 現在の SuperSQL には, テストを作成・実行するためのツールやテストケースが存在しておらず, SuperSQL をオープンソース化するにあたって, SuperSQL の品質確認・動作確認を行うためのテストが必要である. そこで, 本論文では, SuperSQL の開発者がスムーズなテスト実行・管理やテストスイート作成を行うために, SuperSQL 専用のテストツール (SStest) の開発を提案する.

SStest は, GUI による操作によって SuperSQL のテストケース作成・実行・管理を行うツールである. SStest では, データベースにあらかじめ登録されている期待結果と実行結果を比較することで検査を行い, 複数出力される SuperSQL の生成物すべてにおいて検査を行うこともできる. GUI 上から操作が行えるため, SuperSQL の開発者が簡単にテストの実行・追加・削除を行うことができる.

以下, 本稿の構成を示す. まず 2 章で SuperSQL の概要を述べ, 3 章では SuperSQL をオープンソース化するにあたっての問題点を示す. 次に, 4 章では SStest の概要を述べ, 5 章ではテストケース管理・実行支援機能について示す. そして, 6 章では評価について述べ, 最後に 7 章で結論を示す.

2. SuperSQL

この章では, 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, Age

name	age

- 垂直結合子 (!)

データを縦に結合して出力する.

例 : Name! Age

name
age

- 深度結合子 (%)

データを 3 次元方法へ結合する. 出力が HTML ならばリンクとなる.

例 : Name % Age

name	→	age

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

[科目名]! ,[学籍番号]! ,[点数]!

とした場合には各属性間に関連はなく, 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

[科目名! [学籍番号, 点数]!]!

とした場合には, その科目毎に学籍番号と点数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.
クエリ例：[employee.name@{width=200, color=red}]!

2.4 関　　数

2.4.1 link 関数 (出力メディアが HTML の場合のみ)

link 関数は, FOREACH 句と同時に用いる. 深度結合子と同様にリンクを生成することができる. リンクを生成するために深度結合子を用いる場合は全てを一つの質問文で記述するが, link 関数と FOREACH 句を用いる場合はリンク元のページとリンク先のページを別々の質問文で記述する. リンク元を生成する質問文で link 関数を, リンク先を生成する質問文で FOREACH 句を記述する. 以下に例を示す.

クエリ例：

⟨**Q1.ssql**⟩
```
GENERATE HTML
  [e.name % {e.salary, e.syear}]!
FROM employee e
```

これを link 関数と FOREACH 句を用いると, Q2.ssql と Q3.ssql の二つの質問文となる.

⟨**Q2.ssql**⟩
```
GENERATE HTML
  [link(e.name, file="Q3.ssql", att=e.id)]!
FROM employee e
```

⟨**Q3.ssql**⟩
```
FOREACH e.id
GENERATE HTML
  [e.salary, e.syear]!
FROM employee e
```

link 関数ではペアとなる FOREACH 句を含む質問文のファイル名を file= ” ”で指定. さらに, link 関数の att= と FOREACH 句で同じ属性を指定する. この属性の値を利用して URL を生成するので, 指定する属性はリンク先のページを一意に識別できるもの (主キーなど) を選択し, 2 つ以上の属性を指定する場合 (主キーが 2 つ以上あるようなとき) は, link 関数では att1、att2、att3、... と属性を一つずつ指定し, FOREACH 句では属性を (,) で 区切って指定する.

3. SuperSQL オープンソース化での問題点

現在 SuperSQL は, 慶應義塾大学理工学部情報工学科遠山研究室内での研究が進められており, SuperSQL のオープンソース化を目指している. そこで, SuperSQL をオープンソース化するにあたって, SuperSQL の品質確認・動作確認を行うためのテストが必要である. いくら注意しながら開発を行っていても, 全てのプラットフォーム, 全ての環境で SuperSQL の機能全てが正常に動くことは保証はできないためである. さらに, SuperSQL の場合には, オープンソース化した際には, 外部の開発者たちが, SuperSQL のソースコードに手を加えていくことになるため, バグ修正や機能追加などによって既存のソースコードを部分的に変更することになる. しかし, ソースコードに対して, 少しでも変更を行うと, 修正個所とは別の機能が動作しなくなる, デグレードといった現象が生じることがあるため, デグレードの有無を調べるテストである回帰テストを行いながら, 開発を進める必要がある.

しかし, 現在の SuperSQL には, 回帰テストを行うための, テストスイートが存在しておらず, テストスイートを作成する必要があった. また, SuperSQL のテスト実行・作成・管理に適したツールが存在していなかったため, 効率的に SuperSQL のテストを作成・実行・管理するための, SuperSQL 独自のテストツール (SStest) を開発し, さらに SuperSQL のテストスイートを作成した.

本論文で提案する SStest では, SuperSQL の複数の生成物や, SuperSQL の実行が失敗した際に出力されるエラー文章に関しての検査を行うことが可能となっている. また, GUI を用いた操作によって, テストの実行・管理を行い, 開発者がテストをする際の負担を軽減している. この SStest とテストケースによって, 開発者はデグレードの恐れを最小限に留めながら, 開発

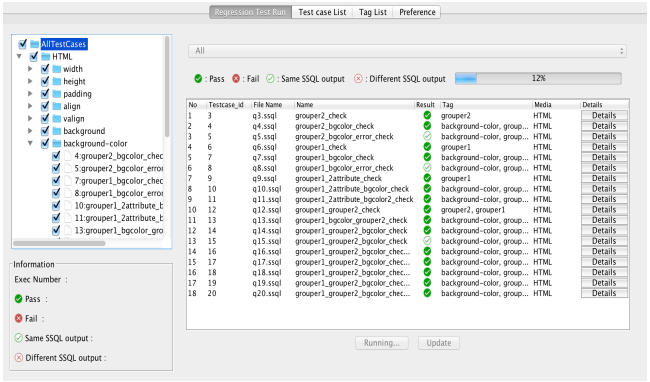

図 1　テスト実行

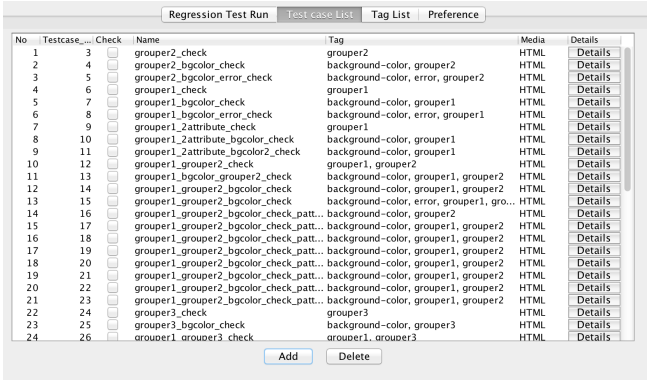

図 2　テストケース一覧

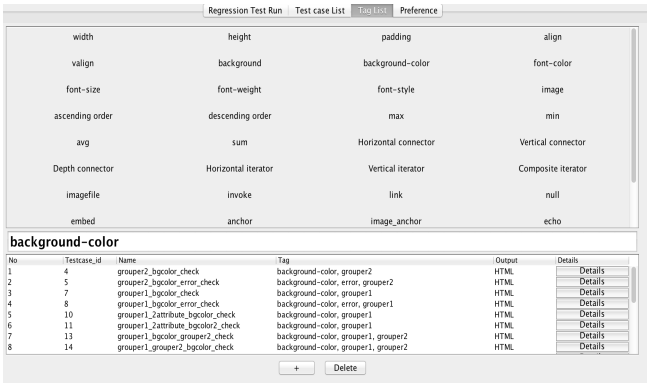

図 3　タグ管理

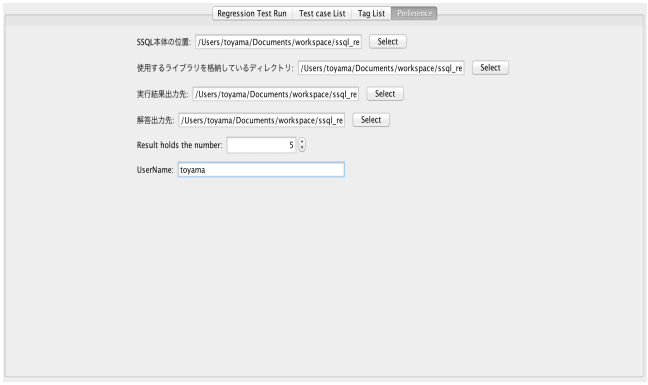

図 4　設　　定

を行うことができる．

4.　SStest

4.1　概　　要

SStest は Java を用いて実装された Java アプリである．SStest では，SuperSQL のテストケースの実行・管理を GUI 上から行うことができる．

4.2　システムの構成

SStest は【TestRun】，【TestCaseList】，【TagList】，【Preference】の 4 つのタブから成り立っている．それぞれを
図 1, 図 2, 図 3, 図 4 に示す．

4.2.1　テスト実行

【TestRun】タブでは，主にテストの実行，テスト実行結果表示を行う．テストの実行を行いたい場合は画面左のツリーにチェックを入れることで実行するテストケースを選択する．そして実行ボタン (Execute) を押すことで，テストが実行され，画面右中央に実行結果が表示される．テスト実行中には進捗バーが現れて，現在のテストの進捗具合がパーセントで表示される．また，テストをタグごとに実行が行えるため，メディアや結合子・装飾子・反復子・関数ごとの実行を可能にしている．画面右上部にはプルダウンが表示されており，実行結果ごとに結果が表示できる．最後に，テストの実行が終了すると，テストの結果が左下に表示される．

テストをタグごとに実行を行う場合を図 5 に示す．このように，チェックを入れるのだが，下の階層のすべてにチェックが入ると，その上の階層にもチェックが入るようになっている．しかし，1 つでも未選択のものがあれば，1 つ上の階層は赤で未選択であることを示している．

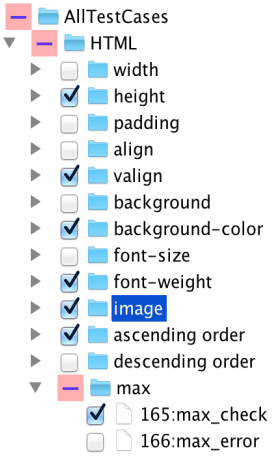

図 5　テストケース選択

4.2.2　テストケース一覧

【TestCaseList】タブでは，登録されているテストケース一覧と表示とテストケースの追加・削除が行えるようになっている．テストケース一覧のチェックボックスにチェックを入れて削除ボタン (Delete) を押すことで不要なクエリは削除を行うことができる．また，各テストケースに関する詳細は右の詳細ボタ

ン (Detail) を押すことで確認することができる．テストケース追加に関しては，画面下部に表示されている追加ボタンを押すことでテストケースの追加画面が表示される．テストケースの追加の入力値を SuperSQL のクエリを使用し，実行結果を期待値として登録を行う．テストケース追加時のクエリを実行した際，実行が成功すれば自動的にブラウザを起動し，生成された HTML を確認することができる．

4.2.3 タグ一覧

【TagList】タブでは，現在登録されているタグ一覧とタグの追加・削除を行うことができる．まず，タグには，結合子・装飾子・反復子・関数名などが登録されている．次に，タグ名のボタンをクリックすることで，そのタグに登録されているテストケース一覧が表示され，先ほどと同様に詳細ボタン (Detail) を押すことで，そのテストケースに関してもさらに詳細の情報をみることができる．

4.2.4 設定/preference

【設定】タブでは，SuperSQL 本体の位置，使用するライブラリを格納しているディレクトリ，実行結果出力先，期待結果出力先，期待結果の保持数，ユーザー名の設定を行うことができる．

4.3 処理の流れ

SStest でのテストケース実行・テストケース追加・テストケース置換処理の流れについて説明を行う．

4.3.1 テストケース実行処理の流れ

テストケース実行処理の流れを図 6 に示す．データベースには事前に，テストケースと対応するテストの期待結果が保存されている．まず，SStest 上でテスト実行が行われ，データベースに接続する．そして次に，登録されている各テストケースに対する最新の期待結果のみを選択し，テストケースとテストの期待結果を取得する．そして，取得したテストケースと期待結果のそれぞれをファイルに出力する．最後に，出力されたテストケースを SuperSQL で実行し，SuprtSQL の実行結果とテストの期待結果の差分を取ることによって検査を行っている．

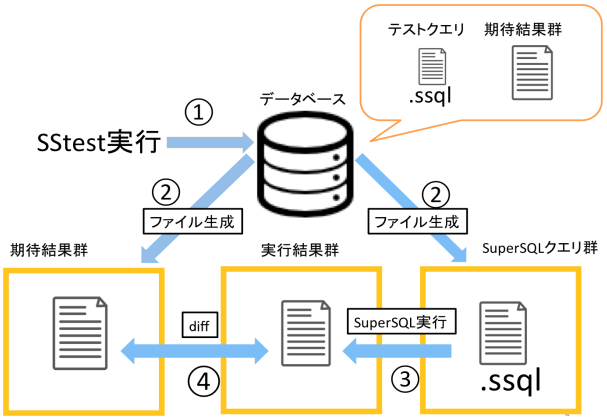

図 6　テスト実行処理

4.3.2 テストケース追加処理の流れ

テストケース追加処理の流れを図 7 に示す．SStest 上で追加ボタンが押されると，入力値として入力されている SuperSQL のクエリが実行される．SuperSQL のクエリが実行され，実行結果が成功・失敗どちらの場合にも，実行結果登録確認が行われる．実行結果登録確認画面で登録された場合に，データベースにテストケース追加される．実行が失敗する場合の登録は，エラー文章の登録を意味する．また，実行が成功する場合の登録は，実行結果の登録を意味する．削除の場合，1 つのテストケースに複数の期待結果が存在している場合があるので，その場合はテストケースに対応したすべてのテスト期待結果の削除が行われる．

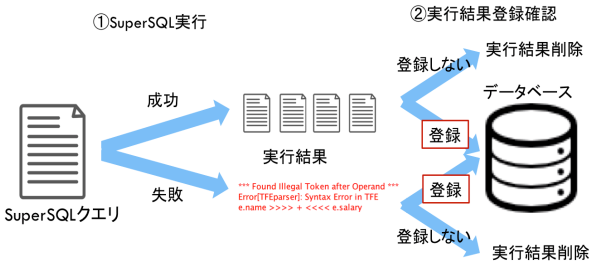

図 7　テストケース追加処理

4.3.3 テストケース置換処理の流れ

テストケース置換処理の流れを図 8 に示す．まず，SStest 上で置換ボタン (replace) が押されると，現在の実行結果が期待結果として登録され，過去の期待結果が期待結果ヒストリーに追加される．SStest では，保持できる期待結果の数が選択可能になっており，期待結果数が最大の場合は一番古い期待結果を削除し，最新の期待結果として登録されるようになっている．また，その他の場合は，最新の期待結果として登録される．

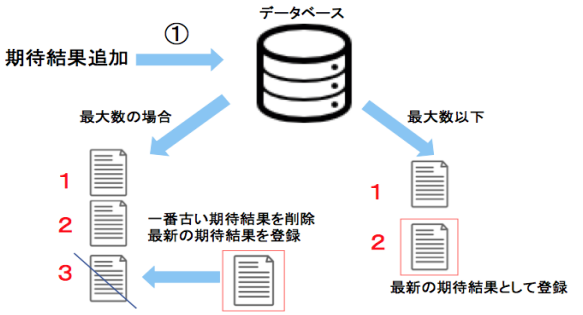

図 8　テストケース置換処理

5. テストケース管理・実行支援機能

SuperSQL 独自の機能に対するテストケース管理・実行支援機能を提供する．これらの機能は，複数メディア対応，SuperSQL の複数の同時生成物に対応，SuperSQL エラー文章検査，テストケース追加時プレビュー，期待結果のヒストリー管理である．

5.1 複数メディア対応

SuperSQL の特徴でもあるように, SuperSQL では様々なメディアを指定することが可能で, メディアによって出力媒体を選択を行っている. そこで, 本テストツールでは様々な出力媒体でも同様のステップでテストの実行・追加を行え, 非常に利便的である. また, テストケースの作成時にメディア選択を行うことになっており, 各メディアごとにテストの実行可能である.

5.2 SuperSQL の複数の同時生成物に対応

SuperSQL の複数の同時生成物の例を図 9 に示す. GENERATE 句の後に HTML と指定することで, 出力媒体を HTML を出力として指定した場合でも, SuperSQL は 1 つのクエリの実行に対して, HTML・JavaScript・CSS など様々な生成物が同時に生じる. このような複数の同時生成物すべてに関して, 一括して検査を行うことができるようになっている. 図 10 のように, フォルダを再帰的に読み, 生成物すべてについて, 対応する生成物の以前との差分を取ることで, すべての生成物に関して, 検査漏れなく微細な変化にも対応可能となっている.

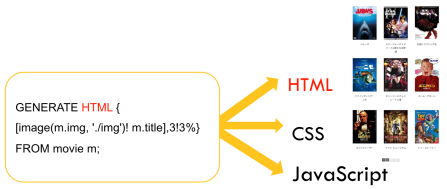

図 9　SuperSQL の複数の同時生成物

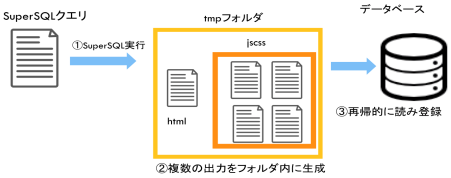

図 10　SuperSQL の複数の同時生成物処理

5.3 SuperSQL のエラー文章検査

SuperSQL のテストケースに関して, SuperSQL の実行が成功する場合の検査と, SuperSQL の実行が失敗する場合の検査を行うことができる. SuperSQL の実行が成功・失敗するどちらの検査に対しても成功・失敗の 2 種類の結果が存在している.

SuperSQL の実行が成功する場合は, 実行によって生成された生成物を検査することになる. SuperSQL の実行が失敗する場合は, 実行失敗時に出力されるエラー文章に関して検査を行う. エラーの出力なども SuperSQL 開発者によって, 研究されており変化していくものなので, 正しいエラーが出力されているのかを検査することが必要であり, エラー文章に関しても検査を行えるような機能をつけている.

5.4 テストケース追加時プレビュー

テストケース追加時に SuperSQL の実行が成功すれば, 自動的にブラウザを起動し, 生成された HTML ファイルを確認することができる. この機能により, 結合子・装飾子・反復子・関数などが正しく機能しているかどうかを目視での動作確認も行えるようになっている.

5.5 期待結果のヒストリー管理

期待結果と実行による生成物との間で差分を取ることで検査を行っており, その差分の結果を表示する. 画面上部左側には, 現在の期待結果が表示され, 画面上部右側には, 実行結果が表示されている. 期待結果と実行結果の間に差分がある部分にはハイライトが表示されるようになっている. そして画面下部の diff には差分の個数, 行数, 差分の内容が表示されるようになっている. 現在の期待結果・実行結果の表示のどちらにも行番号を表示することで, 何行目に差分があるという情報をわかりやすくしている. (図 11)

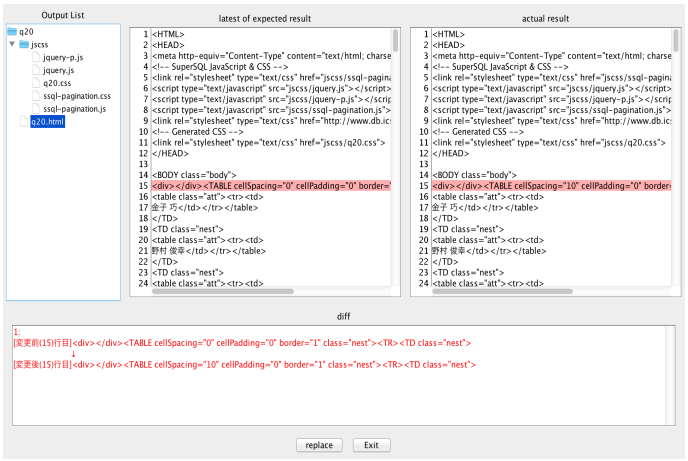

図 11　差分箇所の表示

ここでは, 現在の最新の期待結果と, 実行結果の間に差分があるが, 仕様変更によって変更された可能性も考えられる. 仕様変更によって差分が表示された場合は, 開発者は期待結果を追加することが可能である. replace ボタンを押すことで, 実行結果が期待結果として登録され, 過去の期待結果はヒストリーとして保持される.

6. 評　　価

本研究で提案するテストツール (SStest) の有用性を評価するため, 2 つの方法で評価を行っている. 1 つは, 本研究室で SuperSQL の開発に携わっている方々からの継続的なフィードバック. 2 つ目は特徴ごとの機能比較を行った.

6.1 フィードバック

本論文で提案する SStest は, 慶應義塾大学理工学部情報工学科遠山研究室の SuperSQL 班の方々に, 継続的にフィードバックを行いながら研究開発を行った. 以下に, 開発者による, 継続的なフィードバック例を列挙する.

- SuperSQL のクエリが何をチェックしているのか一目でわかるように
- SuperSQL が正しいエラー文章を出せているか検査をしてほしい
- 誰がいつ仕様変更したかわかるようにしてほしい
- 複数生成物すべてを一括で検査したい
- テストケースを簡単でも GUI 上から管理できるようにしてほしい
- テストケースヒストリー管理できるようにしてほしい
- プレビュー表示もできるようにしてほしい
- タグごとにテスト実行が行えるようにしてほしい
- メイン画面でテストケースが階層的に表示されてるので見やすい
- テストケース管理が視覚的にやりやすい
- 実行結果が一目でわかりやすい

この様な開発者の意見や評価を元に, 開発を行った.

6.2 特徴ごとの機能比較評価

6.2.1 テストケースにクエリを使用する回帰テストとの比較

SuperSQL の回帰テストと同様にクエリを使用している, PostgreSQL [3] の回帰テストとの比較を行った. PostgreSQL はオープンソースのオブジェクト関係データベース管理システムである. PostgreSQL では, SQL 実装についての包括的なテストの集まりを用いて, 回帰テストを行うことによって, ソースのコンパイル後に PostgreSQL が思い通りに動作するかの確認を行っている. 結果を表 1 に示す.

表 1　テストケースにクエリを使用する回帰テストとの比較

機能	PostgreSQL	SStest
実行方法	コマンドライン	GUI
テストケース選択実行	△ (基本, 追加)	○ (タグによる選択)
結果表示	○	○
diff による差分結果表示	○	◎
テストケース一覧表示	○	◎

最初に, 回帰テストの実行方法について, PostgreSQL はコマンドライン上からの実行, SStest は GUI 上からの実行となっている. 次に, テストケース選択実行について, PostgreSQL は, 基本的な回帰テストと, 追加の回帰テストを実行することが可能となっている. 非常に時間がかかる可能性があるテストなどは, デフォルトの回帰テストでは実行が行われないため, 追加のテストといて実行されるようになっている. しかし, SuperSQL では, 選択的にテストケースの実行を行うことも可能で, さらにタグごとに選択的にテストの実行を行えるようにしている. そして, 実行結果の表示に関しては, どちらもテスト実行数・成功数・失敗数が表示されるようになっており同等であった. また, どちらも diff を用いた検証を行っており, テスト結果の表示においても, diff による表示であった. しかし, SStest では GUI 上ということもあるが, diff の箇所をハイライトで表示することも可能で, diff の結果表示において, SStest の方が機能が充実している. また, テストケース一覧を確認する際に, PostgreSQL ではテストケースの確認を行うことは, 可能だが, テストケースの情報やテストケース一覧の確認を行う場合, SStest では GUI 上から簡単に行うことができる.

6.2.2 テスト管理ツールとしての機能比較

SStest はテスト管理・作成・実行を一括して行うためのツールとして開発を行ったため, テスト管理機能についての比較を行った. オープンソースのテスト管理システムである, TestLink [4] と Redmine [5] にテスト管理機能をつけるプラグインである impasse [6] とのテスト管理機能の比較を行った. 結果を表 2 に示す.

表 2　テスト管理ツールとしての機能比較

機能	TestLink	impasse	SStest
テストケース表示	○	○	○
仕様とテストケースの関連付け	○	○	○
タグ (キーワード) を用いた管理	○	○	○
テストケース作成	自動/手動	手動	手動
期待結果の入力	自動/手動	手動	自動
テストケースのバージョン管理	○	○	△
複数出力	△	△	○
実行結果表示	○	○	△

まず, テストケース表示・仕様とテストケースの関連付け・タグ (キーワード) を用いた管理については, すべてのツールにおいて可能であった. 次に, テストケース作成・期待結果の入力方法について, TestLink はその他のテストケース作成を行うツールとの連携によって, テストケースの作成や期待結果の入力を自動で行う機能が備わっている. しかし, それ以外のテストケース作成においては, 手動で行う必要がある. SStest においては, テストケースの作成を行うときに, SuperSQL の出力を自動的に, 期待結果として登録を行うため, 期待結果の入力を手動で行う手間を省くことが可能である.

次に, テストケースのバージョン管理について, TestLink や impasse に関してはテストケースのバージョンの管理を行うことが可能である. SStest においては, テストケースのバージョンの管理ではなく, テストケースのヒストリーの管理を行っている. 各テストケースを誰がいつ変更したかという情報に関して管理を行っているため, テストケース全体でのバージョンを管理しているわけではない. 最後に, 実行結果表示に関して, 他のテスト管理ツールでは, 過去のテスト結果を保持して過去の結果も表示しているが, SStest では, 現在の実行によって出力される結果のみの表示を行っており, 過去のテスト結果も回帰テストのツールには必要であると考えられるため, テストケースのバージョン管理機能, 実行結果表示をさらに充実させる必要がある. しかし, SStest は, SuperSQL の実行によって, 複数の生成されるファイルに関してのテスト作成・実行・管理を単純化することができ, 既存のツールにない独自の機能を提供している.

7. おわりに

本研究では, SuperSQL の開発者向けの専用テストツールとテストスイートを提案し実装を行った. SuperSQL の特徴である１つのクエリに対して生成される複数の生成物への検査, ブラウザ表示の動作確認, テストケース作成, テストケース管理などにより, テスト実行時の時間と手間の短縮やプログラムの仕様の誤解などを防止し, 開発者が開発を行いやすいようなテストツールの作成を行った. テストケース作成の際にテストの説明文を記入させることで仕様を明確にし, 現在の仕様がどうなっているかなど他者も簡単に理解できるようにした. 開発途中にも頻繁にテストを行うことができ, 問題を早期に発見することによる品質向上につながっていくと考えられる.

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] PostgreSQL：http://www.postgresql.org

[4] TestLink：http://testlink.org

[5] Redmine：http://www.redmine.org

[6] impasse：https://www.redmine.org/plugins/impasse