HTMLページ生成時間を考慮したSuperSQLクエリの分割支援ツール

石川　恭子†　　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: †kyoko@db.ics.keio.ac.jp, ††toyama@ics.keio.ac.jp

あらまし　近年、データ集約的ウェブサイトと呼ばれる、巨大なデータベースの内容から構成されるウェブサイトが増加している。このようなウェブサイトでは、データベースから HTML へのマッピングによるページ生成時間が問題となる。慶應義塾大学遠山研究室で開発、研究されている SuperSQL は、関係データベースの問い合わせである SQL の拡張であり、関係データベースからの問い合わせに対する結果を構造化し、HTML などの多彩なレイアウトへの出力を可能とする。一つのクエリの実行によって多段のリンク構造をもつ階層的なハイパーテキストを一括生成できる。本論文では、SuperSQL によるデータ集約的ウェブサイトの生成における、HTML ページ生成時間に着目し、クエリを自動的に分割するツールを開発し、実装、評価する。

キーワード　SuperSQL, 問い合わせ処理, HTML

Query invoking tool for SuperSQL, considering the time of computing HTML pages

Kyoko ISHIKAWA† and Motomichi TOYAMA††

† ††Department of Information and Computer Science, Faculty of Science and Technology,
Keio University
Hiyoshi3–14–1, Kouhoku-ku, Yokohama-shi, Kanagawa, 223–8522 Japan
E-mail: †kyoko@db.ics.keio.ac.jp, ††toyama@ics.keio.ac.jp

Abstract　Recently, the number of web sites with content derived from large databases are increasing, and these web sites are called data intensive web sites. When creating a data intensive web site, an important issue is when to compute the site's pages. SuperSQL is the SQL extensible language, which can specify output structure and media. Execution of one SuperSQL query can create hierarchical hypertext which has many link structure, at once. In this paper, we develop a tool which invokes a query considering the time of computing HTML pages, and evaluate experimentally.

Key words　SuperSQL, Query Processing, HTML

1. まえがき

近年インターネットの爆発的な普及と共に、インターネットと関係データベースを連結した、データベースシステムの利用が注目されている。その結果、データ集約的ウェブサイトと呼ばれる、巨大なデータベースの内容から構成されるウェブサイトが増加している。このようなウェブサイトでは、データベースから HTML へのマッピングによるページ生成時間が問題となる [6]。

SuperSQL は、関係データベースの問い合わせである SQL の拡張であり、関係データベースからの問い合わせに対する結果を構造化し、HTML などの多彩なレイアウトへの出力を可能とする [3]。一つのクエリの実行によって多段のリンク構造をもつ階層的なハイパーテキストを一括に静的に生成できる。[4] においては、ハイパーリンクの参照に質問の呼び出しを対応させ、リンク先のページをデータベースから動的に生成する機能拡張が報告されている。しかし、静的生成、動的生成どちらも大規模データを扱う場合には欠点がある。1000 件程度のデータの簡単なウェブサイトでも、静的生成では HTML ファイルの総量が数 MByte になり、動的生成ではユーザがクリックしてからリンク先のページが表示されるまでに 30 秒近くかかってしまう場合がある。静的と動的を組み合わせることによって、HTML ページ生成時間 (実行時間やユーザの待ち時間) を調整することができる。

本論文では、SuperSQL によるデータ集約的ウェブサイトの生成において、クエリを自動分割するツールを開発、実装、評

価する。、これによって静的と動的を組み合わせ、HTML ページ生成時間を調整する。

2. SuperSQL

SuperSQL はデータベースの内容から出版物を生成することを目的として、SQL のターゲットリストを拡張した言語である [1] [2]。SuperSQL の質問文は、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。目的媒体の指定 < $media$ > は、現在では HTML、XML、Excel、LaTeX などが許されているが、本論文では HTML を対象とする。< 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 結合子と反復子

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る。コン マ (,)、感嘆符 (!)、パーセント記号 (%) の 3 つが、それぞれ第 1〜3 次元に対応する結合子である。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。

反復子は指定する方向に、インスタンス数だけ繰りかえして表示する。一対の角括弧 ([ ]) に上記の結合子を添えたものが、それぞれの次元の反復子である。例えば、

[名前, 学籍番号]!

は、横方向に連結された学生の氏名、学籍番号を、インスタンス数だけ縦方向に連続的に連結する。

また反復子はただ構造を指定するだけでなく、そのネスト関係によってグルーピングの働きを持つ。例えば、

[研究室, [名前, 学籍番号]!]!

は、研究室ごとにグループ化し、それぞれの学生の氏名と学籍番号の一覧が表示される。

2.2 SuperSQL によるハイパーリンクの生成

映画情報データベース (図 1) から以下に示す SuperSQL クエリ Q0 によって得られる WWW ハイパーテキストを図 2 に示す。

```
Q0: GENERATE HTML
    [verb(actor)![p.name%
      {p.name![m.title%{m.title!m.year!p.name}
      ]!}
    ]!]!
```

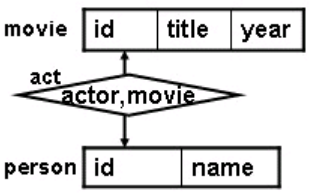

図 1 映画情報データベースのスキーマ

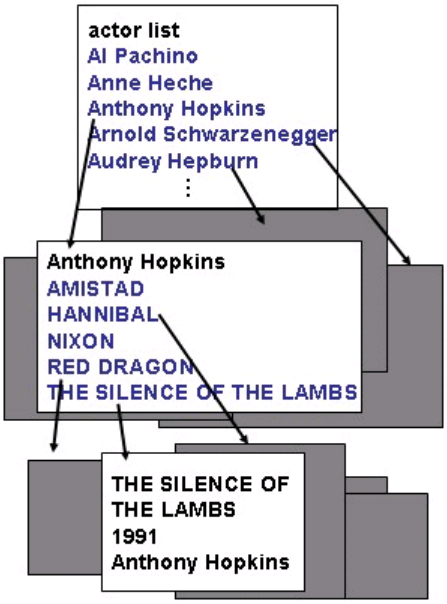

図 2 SuperSQL クエリによるハイパーテキストの生成例

```
    FROM person p, movie m. act a
    WHERE p.id=a.actor AND a.movie=m.id
```

このクエリにより、3 レベルの階層的な WWW ドキュメントが生成される。1 レベル目は 1 ページだけで、俳優のリストを表示する。2 レベル目は選択された俳優の映画のリストを表示し、3 レベル目は映画のタイトル、制作年、主演俳優が表示される。

2.3 invoke 関数による動的質問呼び出し

SuperSQL には、クエリの中から別のクエリを呼び出すために invoke 関数が用意されている。

invoke(file, string1=,att1, string2=,att2, ...)

file は呼び出されるクエリを含むファイルの名前であり、string1=,att1、string2=,att2、... はそれぞれ、現文脈における属性 att の値が string に連結され、呼び出されるクエリの WHERE 句に連言で追加される条件である。

クエリ Q1a、Q1b、Q1c は、前節のクエリ Q0 を invoke 関数を用いて 3 つに分割したものであり、それぞれハイパーテキストの 1〜3 レベル目を生成する。Q0 は、ハイパーテキストの 3 レベルを全て静的に一括生成するのに対し、Q1a、Q1b、Q1c は、Q1a が 1 レベル目を生成し、2、3 レベルの 1 ページを Q1b、Q1c がそれぞれ動的に生成する。

```
Q1a: GENERATE HTML
     [verb(actor)!
     [p.name%invoke(q1b.sql,p.id=,p.id)]!]!
     FROM person p

Q1b: GENERATE HTML
     [p.name!
     [m.title%invoke(q1c.sql,m.id=,m.id)]!]!
     FROM person p, movie m. act a
```

WHERE p.id=a.actor AND a.movie=m.id
(AND p.id=xx) ; dynamically added

Q1c: GENERATE HTML
[m.title!m.year!p.name]!
FROM person p, movie m. act a
WHERE p.id=a.actor and a.movie=m.id
(AND m.id=yy) ; dynamically added

Q1b には Q1a の invoke 関数で分かる通り、選択された俳優の id(例えば xx) を含む条件 (p.id=,xx) が動的に追加される。同様に、Q1c には条件 (m.id=,yy) が追加される。

Q0 と Q1a、Q1b、Q1c の等価性については [5] で述べている。

3. 大規模データを扱うときの従来の問題点

第 2. 章で述べたように、SuperSQL の第 3 次元結合子 (%) によるハイパーリンクの生成方法は、結合子 (%) のみを用いて静的に生成する方法と、invoke 関数を用いて動的に生成する方法の 2 通りある。どちらも大規模データを扱う場合には問題がある。以下で、それぞれの問題について説明する。

3.1 静的生成の場合

結合子のみを用いた場合、全てのページが一括に静的に生成される。(%) で結合された左辺の属性のデータベースに入っているデータ数が多いと、生成される HTML ファイル数も多くなり、その総量は膨大になる。例えば、2.2 節のクエリ Q0 の最初の (%) で、データベースに入っている p.name のデータ数が 1000 件の場合、2 レベル目のページとして静的に生成される HTML ファイルの数は 1000 になってしまう。また、生成される HTML ファイル数が多くなればなるほど、一つのクエリの処理にかかる時間も長くなる。

3.2 動的生成の場合

invoke 関数のみを用いた場合、1 レベル目の 1 ページ以外はユーザがウェブ上である値をクリックすることによってクエリが呼び出され、それを実行してリンク先のページが生成される。この場合、3.1 節で述べたような問題はない。しかし、動的に生成されるページに表示されるデータ数が多いと、そのデータをデータベースから取ってきてソート、レイアウト等にかかる時間が長くなる。

3.3 解決するには

3.1、3.2 それぞれの問題点の逆はもう一方の利点である。両者を組み合わせることにより、それぞれの欠点を補うことができる。組み合わせるといっても、ユーザがどこを (%) で、どこを invoke にするか判断するのは大変である。また判断できたとしても、手動で (%) を invoke にしてクエリを分割するのは骨の折れる作業である。そこで本論文では、結合子 (%) のみを用いて書かれたクエリを、結合子 (%) と invoke 関数を組み合わせた複数のクエリに自動的に分割するツールを開発する。

4. 質問分割

4.1 質問分割で考慮する点

クエリを分割する際に考慮する点として、大きく分けて以下の 3 つを挙げる。

1. ページ作成時間
   - 更新時間 (SuperSQL クエリの実行にかかる時間)
   - 応答時間 (クリックしてから表示されるまでにかかる時間)
2. HTML ファイルの総量
3. データのフレッシュネス

1、2 については第 3. 章で述べた問題となる要素であり、(%) と invoke を組み合わせることによって、調節できる。3 については、全て invoke にするか、(%) の場合はデータベースの内容が更新されたら SuperSQL クエリを実行しなおすことによって、常にフレッシュなデータを得ることができる。本論文では、常にデータをフレッシュな状態にするとして、1、2 を考慮してクエリを分割する。1、2 を考慮して、どこを invoke にするかを決めなければならないが、その判断要素として同レベルの同種ページ数の合計を考える。

4.2 同レベルの同種ページ数の合計

同レベルの同種ページ数の合計を例を使って説明する。以下のクエリを実行すると図 3 のようなのハイパーテキストが生成される。この図を見ると分かるように、2 レベル目は a.name、b.name からの 2 種類のページ A 群、B 群が生成され、3 レベル目は c.name からのリンクのみで、1 種類のページ C 群が生成される。A 群のページは、1 レベル目で表示されている a.name のリストの一つをクリックすると表示されるので、A 群のページ数の合計はデータベースに入っている a.name の数となる。同様に、B 群、C 群びページ数の合計は、データベースに入っている b.name、c.name の数である。以上のことから、同レベルの同種ページ数の合計は、(%) の左辺に書かれている属性のデータベースに入っているデータ数と言える。

```
GENERATE HTML
[ a.name% { … },
  b.name%
    {c.name% { … }}
]!
FROM a, b, c
WHERE …
```

4.3 分割の方法

同レベルの同種ページ数の合計が少ない (数 10 ページ) ときは (%) のままで、多いときは invoke 関数を用いて分割する。この理由を以下で説明する

- 同レベルの同種ページ数の合計が少ないとき
  1 ページに表示されるデータ数が多い場合もあり invoke だと応答時間が長くなってしまうので (%) の方が良い
- 同レベルの同種ページ数の合計が多いとき
  (%) だと更新時間が長くなり、HTML ファイルの総量が大きくなってしまうので invoke の方が良い

分割は以下の手順で行う

1. 与えられたクエリを (%) の部分で全て分割
2. 分割したそれぞれの SQL を実行し、ページ数をカウント

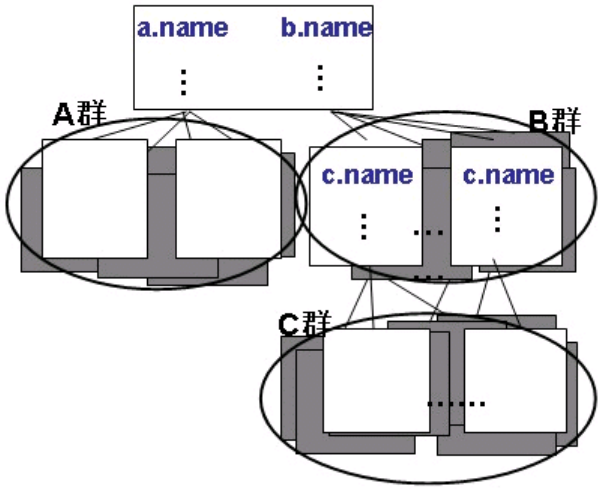

図 3　同レベルの同種ページ

3. ページ数が n 以下なら (%) のまま、n より大きければ invoke
   n は発見的手法によって求める。

5. 実験と評価

自動分割の有用性を以下の 3 つの基準を設けて評価する。

1. 更新時間 [sec](SuperSQL の実行時間)
2. HTML ファイルの総量 [byte]
3. 平均応答時間 [sec](ユーザがクリックしてからリンク先のページが表示されるまでの平均時間)
4. 最大応答時間 [sec](ユーザがクリックしてからリンク先のページが表示されるまでの時間が最大のもの)

自動分割したものと、リンク部分ががが全て (%) のもの (static)、全て invoke のもの (invoke) の 1〜4 を比較する。

実験を行った環境は、CPU Pentium(R)Ⅲ 1.4G × 2、メモリ 2GB、OS Linux(Redhat7.3)、DBMS PostgreSQL 7.3.1 で、インターネットブラウザは Mozilla1.0.1 を用いた。図 4 の映画情報データベースのスキーマを用い、各タプル数が表 1 の 3 通りの場合で、2 つのクエリについて行った。n は 10 とした。

2 つのクエリはそれぞれ図 5 の左右のようなハイパーテキストを生成する。実験 1 では 2 レベル目に producer のリストが表示され、3 レベル目で producer 別映画一覧が表示され

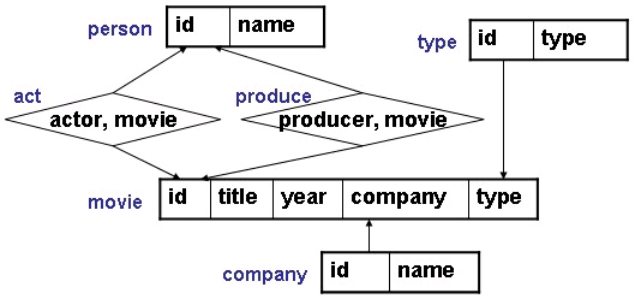

図 4　映画情報データベースのスキーマ

表 1　各テーブルのタプル数

		m1	m2	m3
movie		100	1000	10000
person	actor	20	200	2000
	producer	10	100	1000
company		5	50	500
type		10	10	10

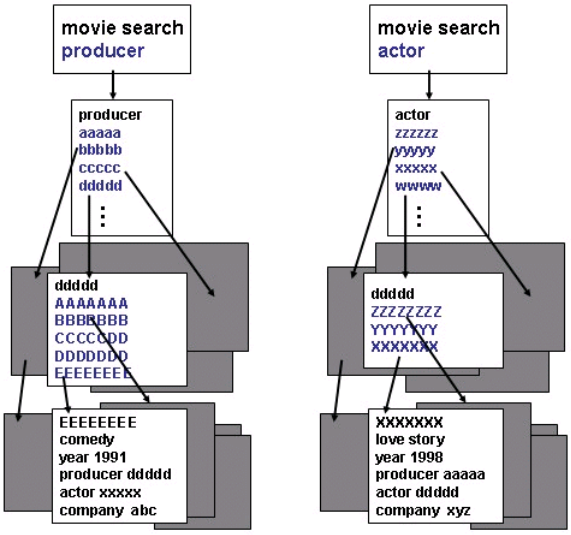

図 5　左：実験 1 のクエリによって生成されるハイパーテキスト
右：実験 2 のクエリによって生成されるハイパーテキスト

る。producer のリストには、表 1 の producer のタプル数だけ producer の名前が表示され、3 レベル目のページ数は producer のタプル数となる。実験 2 では 2 レベル目に actor のリストが表示され、3 レベル目で actor 別映画一覧が表示される。actor のリストには、表 1 の actor のタプル数だけ actor の名前が表示され、3 レベル目のページ数は actor のタプル数となる。このように、生成されるハイパーテキストの構造は似ているが、表示されるデータ数やページ数の異なる 2 つのクエリについて実験を行った。

5.1 実　験　1

まず、図 6 のクエリで実験を行った。このクエリによって生成されるページ数は m1、m2、m3 それぞれ 112、1102、11002 である。自動分割を行ったものは、m1 は 1、2、3 レベル目が静的に、4 レベル目は動的に生成され、m2、m3 は 1、2 レベル目が静的に、3、4 レベル目は動的に生成される。更新時間、HTML ファイルの総量、平均応答時間、最大応答時間は図 7、8 の通りである。

更新時間、HTML ファイルの総量は、invoke は m1〜m3 でほぼ一定であるのに対し、static、st+ivk はタプル数が多いほど増加している。invoke は 1 レベル目の 1 ページのみが静的に生成され、その 1 ページはデータ数に関係ないので一定の値となり、static、st+ivk と比べて小さい値となっている。static

```
GENERATE HTML
[ verb(movie search)! verb(producer)%%
   {verb(producer)! [pp.name%%
      {verb(producer), pp.name}! [m.title%%
         {m.title! {verb(type), t.typename}!
         {verb(year), m.year}! {verb(producer), p1.name}!
         {verb(actor), p2.name}! {verb(company), c.name}}
      ]! }
   ]! }
]!
FROM movie m, type t, person pp, produce pr, person pa,
    act a, company c
WHERE t.id=m.type and m.id=pr.movie and
    pr.producer=pp.name and a.movie=m.id and
    pa.id=a.actor and c.id=m.company
```

図 6　実験 1 で用いたクエリ

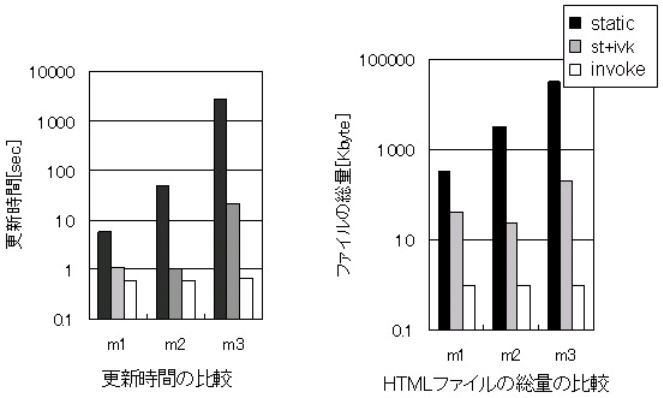

図 7　実験 1：更新時間と HTML ファイルの総量の比較

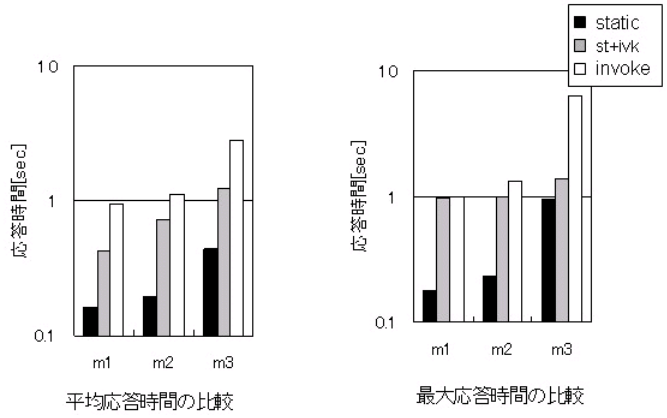

図 8　実験 1：平均応答時間と最大応答時間の比較

はタプル数が増えると、更新時間と HTML ファイルの総量は増加している。st+ivk の更新時間は m1、m2 がほぼ同じであり、これは m1 では 3 レベル目までの 12 ページが静的に生成されるのに対し、m2 は 2 レベル目までの 2 ページしか静的に生成されないからである。静的に生成されるページ数が m2 の方が少ないため、HTML ファイルの総量は m1 より小さい。

st+ivk の更新時間は m3 で static の 130 分の 1、HTML ファイルの総量は 165 分の 1 であり、static と比べかなり良い結果が得られている。

応答時間は static は全て 0.5 秒以下であるのに対し、st+ivk は m3 で 1.2 秒、invoke は 2.8 秒かかっている。しかし、m3 の最大応答時間は st+ivk は 1.4 秒で平均応答時間とあまり変わらないのに対し、invoke は 6.2 秒もかかっている。

4 つのグラフをまとめて見ると、m1、m2 は多少差はあるが、どの要素を重視するかで 3 つのうちどれが良いか異なる。m3 は、st+ivk は static と比べ更新時間、HTML ファイルの総量はかなり良い結果が得られ、応答時間は invoke と比べ良い結果が得られている。このことから、自動分割は有用であると言える。

5.2 実　験　2

次に、図 9 のクエリで実験を行った。このクエリによって生成されるページ数は m1、m2、m3 それぞれ 122、1202、12002 である。自動分割を行ったものは、1、2 レベル目が静的に生成され、3、4 レベル目は動的に生成される。更新時間、HTML ファイルの総量、平均応答時間、最大応答時間は図 10、11 の通りである。

実験 1 と同様、invoke の更新時間と HTML ファイルの総量はほぼ一定である。static、st+ivk の更新時間と HTML ファイルの総量は、タプル数が増えると増加している。st+ivk は 2 レベル目までの 2 ページのみが静的に生成されるので、m3 の更新時間と HTML ファイルの総量を static と比べると、それぞれ 53 分の 1、136 分の 1 であり、かなり良い結果が得られている。

static と st+ivk の平均応答時間は、m1、m2 は 1 秒以下だが、m3 では 1.3 秒と 1.8 秒である。これは、m3 の最大応答時間が static、st+ivk どちらも 3 秒ほどかかっているためである。3 秒かかった理由は、そのページの HTML ファイルのサイズが 375Kbyte と大きいためである。ただし、3 秒というのはページ全体が表示されるのにかかった時間であり、ページの一部は 1 秒位内に表示されている。

invoke の平均応答時間と最大応答時間は、m3 で 42 秒と 2 分である。これは、2 レベル目で 2000 件の俳優を表示するためにかかる時間のためである。

4 つのグラフをまとめて見ると、実験 3 同様 m1、m2 は多少差はあるが、どの要素を重視するかで 3 つのうちどれが良いか異なる。m3 は、st+ivk は static と比べ更新時間、HTML ファ

```
GENERATE  HTML
[ verb(movie search)! verb(actor)%
   {verb(actor)! [pa.name%
      {verb(actor), pa.name}! [m.title%
         {m.title! {verb(type), t.typename}!
         {verb(year), m.year}! {verb(producer), p1.name}!
         {verb(actor), p2.name}! {verb(company), c.name}}
      ]! }
   ]! }
]!
FROM movie m, type t, person pp, produce pr, person pa,
    act a, company c
WHERE t.id=m.type and pr.movie=m.id and
    pp.name=pr.producer and m.id=a.movie and
    a.actor=pa.id and c.id=m.company
```

図 9　実験 2 で用いたクエリ

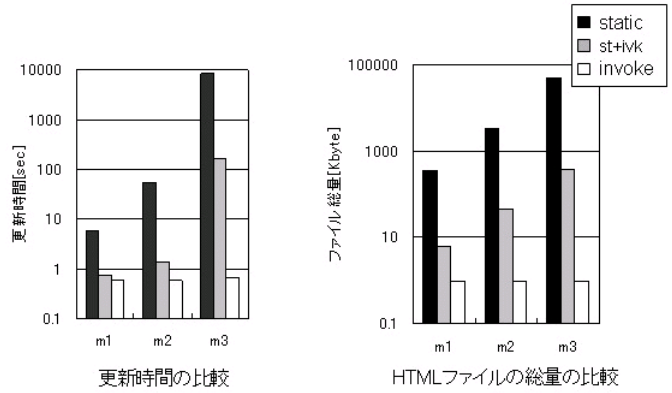

図 10　実験 2：更新時間と HTML ファイルの総量の比較

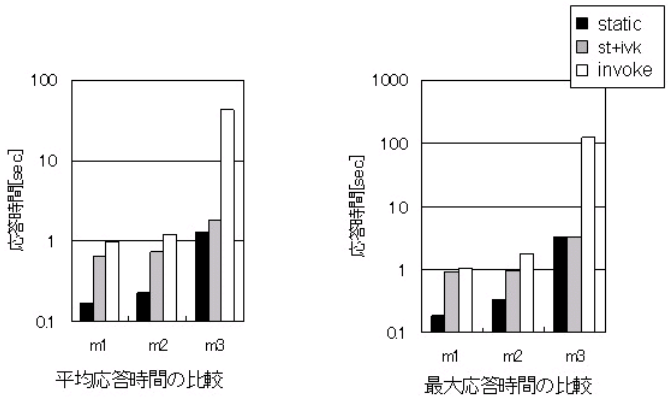

図 11　実験 2：平均応答時間と最大応答時間の比較

イルの総量はかなり良い結果が得られ、応答時間は invoke と比べかなり良い結果が得られている。このことから、実験 2 においても自動分割は有用であると言える。

5.3 評　　価

実験 1、2 において、st+ivk の更新時間は static より短く、HTML ファイルの総量も static より小さかった。また、st+ivk の平均応答時間、最大応答時間は invoke より短かった。

- タプル数が少ないとき

実験 1、2 の両者において static、invoke、st+ivk の差が小さく、3 つのうちどれが良いかはどの要素を重視するかによって異なる。

- タプル数が多いとき

m3 において、実験 1、2 で st+ivk の更新時間は static の 130 分の 1、53 分の 1 であった。HTML ファイルの総量は st+ivk は static の 12 分の 1〜1900 分の 1 であり、static と比べかなり小さかった。また、応答時間は static と st+ivk はあまり差がなく 1 秒程度であるが、invoke は static、st+ivk に比べかなり遅く実験 1〜4 で最大で 3 分であった。自動分割によって、static より更新時間と HTML ファイルの総量を小さくでき、invoke のときの応答時間のワーストケースを回避することができた。

以上のことから、本研究の目的は SuperSQL によるデータ集約的ウェブサイトの生成において、クエリを自動分割することによって静的と動的を組み合わせ、HTML ページ生成時間と HTML ファイルの総量を調整することであり、この目的に対し、自動分割は十分に機能していると言える。

6. まとめと今後の課題

本論文では、SuperSQL によるデータ集約的ウェブサイトの生成において、その HTML ページ生成時間に着目し、クエリを自動的に分割するツールを開発した。

SuperSQL の 3 次元結合子によるハイパーリンクの生成方法には、結合子 (%) のみを用いて全てのページを静的に生成する方法と、invoke 関数を用いて 1 レベル目の 1 ページ以外を動的に生成する方法の 2 通りある。しかし、大規模データを扱う場合、静的生成では、SSQL 実行時間が長くなり、HTML ファイルの総量も大きくなってしまい、動的生成ではユーザがクリックしてからリンク先のページが表示されるまでの時間が長くなってしまう、という問題があった。

この問題点を改善するために、本論文では同レベルの同種ページ数の合計を基準として、同レベルの同種ページ数が少ないときは結合子 (%) を用い、多いときは invoke 関数を用いるということにより、結合子 (%) のみで書かれた SuperSQL クエリを結合子 (%) と invoke 関数を組み合わせてクエリを分割する手法を提案した。

この分割手法を実装し評価実験を行った結果、大規模データを扱う場合、自動分割を行ったものは、

- 結合子 (%) のみより、更新時間は短くなり、HTML ファイルの総量はかなり小さくなった
- invoke 関数の場合の、応答時間のワーストケースを回避することができた

以上のことから、本論文で提案、実装した自動分割の結果は十分に機能していることが確認された。

本論文では、発見的手法によって n を求め、その n に基づいて自動分割したに過ぎない。今回の実験で用いた属性のタプル数は、10、100、1000 というように大きな差があったが、例えば全ての属性のタプル数が 20〜25 の間の場合発見的手法によって n を求めるのは容易でない。様々な条件で実験を行い、最適の n を自動的に求め自動分割することが今後の課題である。また、本論文では常に全ての結果を生成することを前提とした。しかし、これは巨大なデータを用いる場合には時間がかかるので、サンプリングなどであらかじめデータを絞り、生成される結果を例示し、その形を確認したら、バックグランドで生成するなどの、処理方式を検討している。

謝　　辞

本研究において、慶應義塾大学理工学研究科・川島英之先輩の御協力を頂きここに感謝致します。

文　　献

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

[2] Motomichi 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] T. Seto, T. Nagafuji and M. Toyama, "Generating HTML Sources with TFE Enhanced SQL," *Proc. ACM Symposium on Applied Computing(SAC '97)*, pp. 96-105, 1997.

[4] 永藤, 瀬戸, 遠山, "TFE による HTML ソースの動的生成," DEWS , pp. 37-42, 1996.

[5] 遠山 元道, "TFE による HTML 生成質問における質問分割の効果と等価性," ADBS，東京，pp. 87-94, 1997.

[6] Daniela Florescu, Alon Levy, Dan Suciu, Khaled Yagoub, "Optimization of Run-time Management of Data Intensive Web Sites," *Proceedings of the 25th VLDB Conference*, Edinburgh, pp. 627-638, 1999.