社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2003－DBS－131 (37)
2003／7／17

SuperSQL 処理系における INVOKE 関数に対するキャッシュ機構

有澤達也[†,††] 石川恭子[†] 遠山元道[†††]

SuperSQL 処理系における HTML 出力では，INVOKE 関数から生成されるリンクを用いることにより，データベースへの動的な評価結果をレイアウトして取得することが可能である．このリンクをたどる度に，Web サーバでは SuperSQL 処理系を呼び出し，データベースへのアクセスを行う．しかし，Web ナビゲーションとして INVOKE 関数を用いた場合，同一条件での再取得が生じる可能性が高い．

そこで本論文では，SuperSQL 処理系における INVOKE 関数を用いた動的な HTML 生成要求に対して，生成結果のキャッシュを行う手法について提案する．

Data caching for INVOKE function of SuperSQL system

Tatsuya Arisawa,[†,††] Kyoko Ishikawa[†]
and Motomichi Toyama[†††]

SuperSQL system is used for generating various HTML documents from relational database. It generates dynamic evaluation result to a database by using links generated from INVOKE function. Whenever this link is clicked, SuperSQL system is executed in a Web server, and access database. However, it happens frequently to generate the same HTML document because of the same condition in Web navigation.

In this paper, we propose caching system for SuperSQL which caches HTML documents generated dynamically by SuperSQL INVOKE function, and use the cache for the same request later.

1. はじめに

現在，Web での情報発信の手段として，ユーザからの要求時にバックエンドにあるデータベースから動的に情報を取得し，その結果から Web ページを作成して提供することがある．例えば，飲食店の情報を同一のデザインを用いて表示する際に，店の情報のみをデータベースに格納し，実際に参照される際に動的に生成して提示する等である．

慶應義塾大学遠山研究室で開発している SuperSQL[1),2)] では，関係データベースからの検索結果をグルーピング等指定した通りのレイアウトで HTML を生成することができる．SuperSQL 処理系には，このような動的に Web ページを生成するために，INVOKE 関数が用意されている．INVOKE 関数で作成されたリンクには，リンク先で生成すべき SuperSQL 質問文および検索条件が埋め込まれており，リンクをたどる都度に，CGI を用いて SuperSQL 処理系にデータを渡し，動的に HTML を生成してユーザに提示を行う．

このようにして作成されたナビゲーションを目的とした Web サイトでは，前のページに戻る等の操作によって同一ユーザが何度も同一内容のページを取得する必要があることが多い．もし，その間にデータベースに格納されているデータに変更があるならば，前回取得時と異なる HTML を生成する必要があるが，データに変更がない場合は前回取得時と同一の HTML を生成することになる．

現在，CGI 経由で要求された SuperSQL 質問文は，格納されているデータの変更の有無に関係なく，常にデータベースへの問い合わせを行っている．このため，同時に要求を受けたり SuperSQL 質問文が複雑になると，サーバでの処理に時間がかかり，ユーザへの提示が遅れてしまっていた．

そこで本論文では，SuperSQL 処理系において，IN-

† 慶應義塾大学大学院 理工学研究科 開放環境科学専攻
School for Open and Envionmental Systems, Graduate School of Science and Technology, Keio University.
†† 慶應義塾大学インフォメーションテクノロジーセンター本部
Keio University Infomation Technology Center.
††† 慶應義塾大学 理工学部 情報工学科
Department of Information and Computer Science, Faculty of Science and Technology, Keio University.

VOKE 関数を用いた動的な HTML 生成要求に対して，生成結果のキャッシュを行うシステムについて述べる．つまり，一度要求のあった生成結果をサーバ側でキャッシュすることで，同一要求に対する応答時間の短縮を目指す．

2. SuperSQL と HTML 出力

2.1 SuperSQL

SuperSQL は，関係データベースへの問い合わせと同時に，その検索結果の構造化を行い，出力を行う処理系である．この SuperSQL の中心となるものは，関係データベースから情報を抽出するための操作言語 SQL のターゲットリストの拡張 TFE であり，階層構造情報と装飾情報を同時に表すことができる．

SuperSQL 質問文の記述方法は以下の通りである．この中で From 句および WHERE 句の持つ意味は，通常の SQL の持つ SELECT 文と同じである．

GENERATE	*medium*	*TFE*
FROM	*from_clause*	
WHERE	*where_clause*	

medium では，検索結果をどの種類の応用データとして出力を行うかを指定することができ，HTML を指定すれば HTML ドキュメントを生成することができる．

また，TFE (Target form expression) は通常の SQL のターゲットリストにでてくるデータベース属性に対して，結合子によってレイアウト方向を指定したり，反復子によるデータのグルーピング，そして「@{}」により装飾情報を付加することができる．

結合子は属性等を「,」(横)，「!」(縦)，「%」(深さ)，「#」(時間) で区切ることによって，それぞれの方向にレイアウトすることを意味する．また，SuperSQL の持つ強力な表現力の基となるグルーピングは，反復子を用いて記述する．反復子は，反復させたい部分を '[ ]' で囲み，その直後に反復方向を結合子と同じ記号で記述することで指定する．

特に HTML を生成する場合，連結子に「%」を指定した場合は，リンク方向への結合を表し，前の属性に対して後ろに続く TFE に対するリンクを生成することになる．

例えば，

```
title % [ actor ]!
```

といった TFE の場合は，`title` の文字列の所にリンクが生成され，そのリンクをたどるとその `title` に関連する actor の一覧を得ることができるということである．

2.2 HTML の静的生成と動的生成

これまで，SuperSQL 質問文を用いて HTML ドキュメントを生成できるということを述べた．Web サイトでは複数のページによって構成されているため，実際には SuperSQL 処理系を用いた Web サイト構築には，大きくわけて，静的生成と動的生成の二通りのアプローチが存在する．

静的生成は，Web サイト全体を生成する SuperSQL 質問文を用いて，予め全ての Web ページを生成する方法である．このため，ユーザからの要求に対して迅速に生成結果を応答することができる．しかしながら，全体を網羅するような SuperSQL 質問文は長く複雑なものになるため，手動で作成するには相当のスキルを必要とする．また，関係データベースに対する問い合わせ時に複雑な結合を伴うため，生成時に実行時間が大きくなってしまうことが多い．そして，データベースの変更に対しては，全体の再生成を行わない限り反映することができないという欠点を持つ．

これに対して，動的生成ではページ単位で SuperSQL 質問文を用意し，INVOKE 関数を用いて生成したリンクによってページ間の関係を定義する方法である．INVOKE 関数で生成されるリンクは CGI 経由で SuperSQL 処理系を呼び出し，その時点でのデータベースにアクセスし HTML を生成する．動的に作成するため，常に最新のデータベース情報から生成された結果を評価することができる．しかし，サーバへのアクセス時に常に SuperSQL 処理系が起動されるため，要求が集中した際にサーバへの負荷が大きくなってしまう．

4) は，静的生成では SuperSQL 処理系での処理にコストのかかる SuperSQL 質問文に対して，部分的に INVOKE 関数を用いた動的生成に変更することにより，処理系での生成コストの削減を行っている．この方法では，グルーピング単位で新たな SuperSQL 質問文を生成し，この質問文へのリンクを INVOKE 関数で生成することによって，実現している．

SuperSQL を用いた Web サイト構築において，INVOKE 関数を用いた動的生成は欠かせないものとなる．この INVOKE 関数について次節で詳しく述べる．

3. INVOKE 関数

3.1 INVOKE 関数の書式

SuperSQL では WWW ナビゲーション機能を実現するために，TFE 内に INVOKE 関数を用いる．IN-

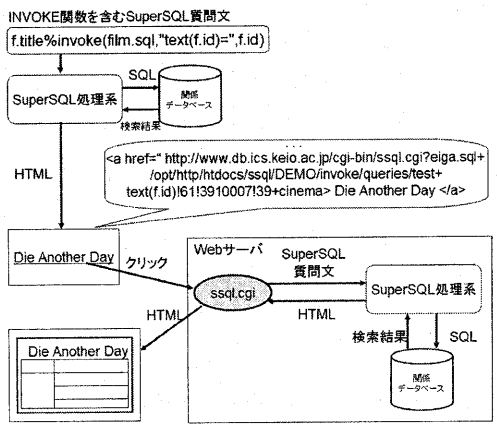


図 1　INVOKE 関数を用いた動的な HTML ページ生成

VOKE 関数は，リンクで SuperSQL 質問文の where 句の条件を属性値に応じて付加しながら CGI 経由で SuperSQL 処理系を呼び出し，動的に生成する方法である．

INVOKE 関数の書式は以下の通りである．

att % INVOKE(*query_file*, *selection_condition*, *selection_attribute*)

INVOKE 関数内の引数については，動的に呼び出す SuperSQL 質問文のファイル名を示す *query_file* と，その SuperSQL 質問文に付加する検索条件に用いるための条件文 *selection_condition* と，その条件となる値を決定するためのデータベース属性名 *selection_attribute* を指定する．

また，INVOKE 関数は属性へのリンクとして生成されるために，関数の前に「%」を用いてリンク元の文字列等を表す属性（*att*）が必要となる．

3.2 INVOKE 関数の処理

INVOKE 関数を用いた動的な HTML ページ作成の処理の流れについて，図 1に示す．

まず，呼び出し側を生成する SuperSQL 質問文で 3.1 節の書式で書かれた INVOKE 関数は，SuperSQL 処理系によって，次のようなリンクが生成される．

 value

「%」の前に与えられた属性を評価した結果に対して *value* が生成され，この文字列に対して ssql.cgi へのリンクを生成している．ssql.cgi は INVOKE 関数から SuperSQL 処理系を呼び出すための CGI プログラムになっており，動的に生成するために必要な情報を「?」以下の環境変数 QUERY_STRING として与えている．

condition は INVOKE 関数で指定した検索条件 *selection_condition* にデータベースから実際に取得した属性名 *selection_attribute* を連結したもので，この条件が *query_file* で指定された SuperSQL 質問文の付加される条件となる．

このほか，呼び出す SuperSQL 質問文の path を示す *query_path* とデータベース名を表す *db_name* が指定される．

リンク経由で呼び出された ssql.cgi では，*query_path* と *query_file* で示されたファイルの SuperSQL 質問文に，*condition* で指定された条件を付加した SuperSQL 質問文を生成し，バックエンドの SuperSQL 処理系に問い合わせを行う．この問い合わせ結果をユーザに返すことになる．

また，この呼び出される SuperSQL 質問文にさらに INVOKE 関数を含めることによって，この生成ページの関連ページへのリンクを作成することが可能になる．例えば，映画情報に関してナビゲーションできる Web サイトの SuperSQL による構築の例は図 2のようになる．

3.3 従来の INVOKE 関数処理の問題点

INVOKE 関数によって実行される CGI プログラム ssql.cgi は，要求の度に SuperSQL 処理系を呼び出しており，SuperSQL 処理系ではその都度関係データベースへのアクセスを行う必要がある．

呼び出される SuperSQL 質問文が複雑な場合は，SuperSQL 処理系でのデータベースへのアクセスや構造化等にかかる処理時間が長くなってしまうため，全体として HTML を取得するまでの時間が長くなってしまう．また，アクセスが集中した場合サーバでの処理能力にしたがって，結果を取得するまでに遅延が生じてしまうことになる．

特に Web ナビゲーションを前提にした場合，同一のユーザが巡回している際に前のページに戻る等の操作のため，同一の CGI 呼び出しを必要とする場合が発生する．つまり，前回と同一の SuperSQL 質問文を短期間に何度も処理する可能性があるということである．この短期間にデータの更新がなければ，SuperSQL 処理系で生成される HTML は前回の HTML と同一であるが，要求がある度に SuperSQL 処理系を呼ぶ必要があった．

4. キャッシング

前節で述べたように，従来の INVOKE 関数で呼び

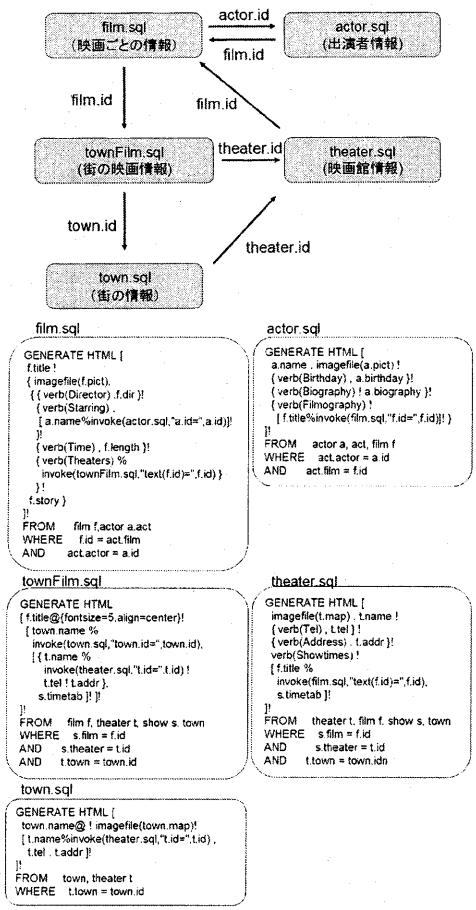


図 2 SuperSQL を用いた Web ナビゲーション例

出された CGI プログラムでは，同一の SuperSQL 質問文に対して繰り返し SuperSQL 処理系を実行することで HTML を生成している．しかし，必要とする SuperSQL 質問文内で結合されている表のタプルが更新されていないならば，同じ SuperSQL に対する答えは同一になる．

このことに着目し，本論文では INVOKE 関数によって生成された CGI へのリンクに対して，SuperSQL 処理系における生成結果をサーバにキャッシュする手法を提案する．この手法を用いることで，一度 SuperSQL 処理系で生成したことのある SuperSQL 質問文が再び与えられた場合，SuperSQL 処理系を利用せずにキャッシュに蓄えられている結果を迅速に提示することが可能になる．

サーバ側で SuperSQL 処理系の結果となる HTML ドキュメントををキャッシングするにあたって，考慮すべき要素として以下のことが考えられる．

- キャッシュした HTML の格納先
- キャッシュのメタデータの格納
- キャッシュの更新

これらの点について，以下で述べていく．

表 1 キャッシュのメタデータ

cached_file	生成された HTML のファイル名
create_time	HTML ページを生成した時刻
access_time	最後に参照された時刻
dbname	SuperSQL 質問文の対象データベース
query_file	SuperSQL 質問文のファイル名
condition	SuperSQL 質問文の WHERE 付加条件

4.1 キャッシュした HTML の格納先

SuperSQL 処理系で生成された HTML ドキュメントの格納先として，メモリ，ファイル，データベース等が考えられる．Web ページのキャッシュであるため，

- サイズがある程度の量になる
- キャッシュからの迅速なデータ取得

ことを考慮すると，ファイルの形で HTML ドキュメントを保持するのが適当だと考えられる．

また，従来の動的生成での SuperSQL 処理系からの処理結果はファイルで渡されるため，そのまま利用できることになる．

4.2 キャッシュのメタデータの格納

要求された INVOKE の条件がキャッシュに存在するかを判定するために，一度処理されたデータについてのメタ情報は，一旦メタデータベースに格納を行う．

メタデータベースでは，生成された WEB ページのファイル名である *cached_file* とともに，表 1の情報を格納する．

この中で，データベース名である *dbname* と SuperSQL 質問文を表す *query_file*，そして WHERE 句の付加条件となる *condition* は，生成される HTML のキーとなる属性である．これら 3 つが等しい場合に限り，このキャッシュされた HTML を用いることができる．

また，最新アクセス時間を表す *access_time* はキャッシュの更新の判定を行うための情報となる．これについては，次節で述べる．

4.3 キャッシュの更新

キャッシュされた HTML の有効性を考えた場合，次の 2 つの項目を確認する必要がある．

- SuperSQL 質問文に変更があった場合
- SuperSQL 質問文で参照されている表が更新された場合

SuperSQL 質問文に変更があった場合は，その SuperSQL 質問文を用いて作成した HTML は全て利用

できなくなる．

また後者のように，トリガ等でデータベースの更新を確認した際には，その更新された表を用いて生成された HTML が全てが影響を受ける可能性がある．SuperSQL 処理系からデータベースへのアクセス結果に差異があるかもしれないからである．差異を調べるにはデータベースへの再アクセスが必要になり，またその結果をキャッシュが有効な間保持しなければ比較ができないため，関連する HTML のキャッシュをすべて無効にするのが適当である．

このようにして，キャッシュ内の HTML ファイルが無効になった場合，これらのキャッシュに対する扱いとして，破棄もしくは更新を行うことが必要である．破棄を行う場合，メタデータベース内の当該タプルも同時に破棄し，HTML を削除する．一方更新の場合は，SuperSQL 処理系に同じ条件で再度生成し，それで得られた HTML ファイルを新たなキャッシュをとして保持する方法である．破棄の方針の方がかかるコストは小さいが，更新の方針の場合，要求のない時間にバックグラウンドで徐々に更新を行うことで，次に同じ条件で生成を要求された場合に，SuperSQL 処理系を呼び出す必要がなくなる．

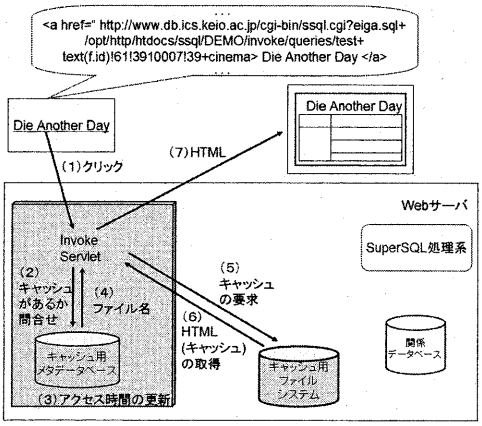
また，キャッシュした HTML 領域が許容容量を越えた場合には，キャッシュのメタデータベースを検索することで，最もアクセスされていないキャッシュファイルから順に削除していくことで，対応できる．

5. 実　　装

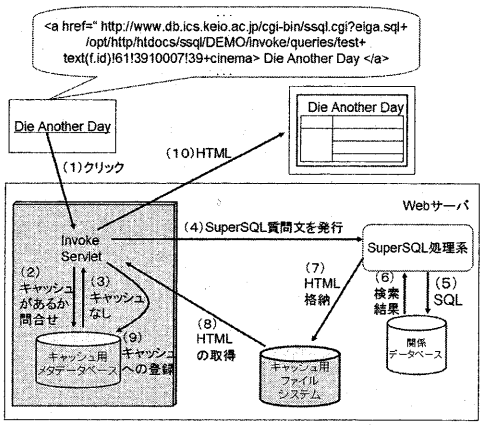
本論文で提案した方式を用いて実装を行った．実装は，既存の動的生成に用いていた CGI プログラムである `ssql.cgi` を置き換える形で行った．つまり，SuperSQL 処理系自身には手を加えず，`ssql.cgi` にあたる SuperSQL 処理系を呼び出すプログラム部分にキャッシュの実装を行った．

SuperSQL 処理系では，入力として SuperSQL 質問文をファイルで与えると，その結果をファイルとして特定のディレクトリに生成する．したがって，生成された HTML をキャッシュするためには，呼び出した際の条件とファイル名の組を記録すればよいことになる．

このキャッシュ機構の実装には Java Servlet[5] を用いた．Java Servlet では，一度呼び出したクラスをサーバに常駐させることにより，永続的なデータ保持を行うことができる．この永続的な部分にキャッシュしたメタデータを格納することで，別の動的生成のリクエストに対してそのメタデータを参照することができる．キャッシュされた Web ページ自身は，ファイル



(a) キャッシュに HTML が取得されている場合



(b) キャッシュに HTML が取得されていない場合

図 3　Java Servlet による実装と処理の流れ

として Web サーバ内に格納され，キャッシュのメタデータベースにファイル名を格納した．また，クラスがアンロードされる際には，PostgreSQL に対して保持しているキャッシュのメタデータを格納する．この格納されたデータをクラスが再度ロードされた際にメモリ内に読み込むことでメタデータベースの永続性を保証する．

また今回の実装では，表の更新等でキャッシュされた HTML が無効になった際には，キャッシュを破棄する方法で実装を行った．

実装した Java Servlet のシステムの概要および処理の流れを図 3に示す．

INVOKE 関数によって生成されたリンクによる要求は，Java Servlet としてサーバで起動している InvokeServlet に伝えられる．InvokeServlet では条件部分をキーとして Java のインスタンスとして実装され

```
GENERATE HTML
 [f.title!
  { imagefile(f.pict),
   {{ verb(Director),f.dir } !
   { verb(Starring),
     [ a.name
       %invoke(actor.sql,a.id=,a.id) ]! } !
   { verb(Time),f.length } !
   { verb(Theaters)
     %invoke(cinema.sql,"text(f.id)=",f.id) }
   } ! f.story
  }
 ]!
FROM  film f,actor a,act
WHERE f.id=act.film and
      act.actor=a.id
```

図4 実験に使ったSuperSQL質問文

ているキャッシュデータベースからキャッシュとして既にHTMLを生成しているかを確認する．

既に生成してある場合には，キャッシュしてあるHTMLのファイル名を取得することができる．InvokeServletはそのファイルを読み込み，ユーザに返す．また，その際にキャッシュデータベースのアクセス時間を更新する．

キャッシュに存在しなかった場合，SuperSQL処理系を呼び出し，HTMLファイルを動的に生成する．そして，この結果生成されるHTMLファイルを読み込み，ユーザに返す．また，同時にHTMLファイル名をキャッシュデータベースに登録を行う．

6. 評価・検討

本論文での提案手法の効果を確認するために，評価実験を行った．評価は，キャッシュを考慮したシステムにおいて，キャッシュに存在する条件が与えられた場合と，キャッシュに存在しない条件でSuperSQL処理系を呼び出した場合，そして従来と同様にキャッシュを全く考慮しない手法について比較を行った．そして，リンクでサーバ側で呼び出された瞬間からサーバで結果を提示する瞬間までの時間を測定した．評価に用いた呼び出される側のSuperSQL質問文は図4である．この質問文に対して，INVOKEのリンクに`f.id`の値を指定して呼び出している．

測定を行った結果は表2の通りとなった．
表2から，キャッシュを用いたINVOKE関数に対する動的生成は，SuperSQL処理系を通して生成を行う処理より高速に結果を返していることがわかった．このことからキャッシングを行うことが非常に効果的であることがわかる．

また，キャッシュにない場合でSuperSQL処理系を呼び出した場合，従来手法より約20msec多く時間がかかっていることがわかる．これは，キャッシュ用メタデータベースへの検索と生成結果の登録にかかった時間である．実装においてキャッシュ用メタデータベースをメモリ内に配置したため，キャッシュ処理によるオーバーヘッドを減らすことができている．

表2 実験結果

提案手法	キャッシュにある場合	1.6
	キャッシュにない場合	163.8
従来手法		143.2

単位: msec

7. おわりに

本論文では，SuperSQL処理系におけるINVOKE関数を用いたHTMLの動的生成の際に，サーバ側で生成結果をキャッシュを行う手法について述べた．一度SuperSQL処理系から生成したHTMLファイルと保持し，その情報をメタデータベースに記録することにより，同一の条件での動的生成に対して迅速にHTMLを返すことができた．

今後の課題として，SuperSQL処理系と融合したキャッシュの構築を行うことが挙げられる．SuperSQL質問文作成の試行錯誤の過程では，同じ構造を持ちながら別のレイアウト，もしくは別のターゲットメディアを生成しなければならないことがあるが，SuperSQL処理系の構造化した時点での中間結果もキャッシュの対象とすることにより，繰り返しのデータベースへのアクセスを抑制することができると考えられる．

参考文献

1) Motomichi Toyama: SuperSQL: An Extended SQL for Database Publishing and Presentation, *Preceedings of ACM SIGMOD '98 International Conference on Management of Data*, pp. 584-586 (1998).
2) SuperSQL HOME PAGE, http://www.db.ics.keio.ac.jp/ssql/index.html
3) C. Agrawal, J. Wolf, P. Yu: Caching on the World Wide Web, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 11, No.1, February 1999.
4) 石川恭子, 遠山元道: 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, データ工学ワークショップ（DEWS2003）, 2003.
5) The Java Apache Project, http://java.apache.org/