SuperSQL の INVOKE 処理における中間データのキャッシュ

Caching Intermediate Result for INVOKE Function of SuperSQL System

有澤 達也♥ 石川 恭子♦
遠山 元道♠

Tatsuya ARISAWA Kyoko ISHIKAWA
Motomichi TOYAMA

SuperSQL 処理系の INVOKE 関数を用いると，動的にデータベースを参照し HTML 文書を生成することができる．以前，生成結果の HTML 文書をキャッシュする手法を提案し，同一質問に対する生成コストを軽減することができるようになった．しかしながら，同一データに対してレイアウトだけ異なった結果が必要な場合，キャッシュを利用できなかった．本論文では，HTML 文書に変換する前段階のデータをキャッシュし，要求された質問文の構造を比較することにより，適切なキャッシュを用いて類似質問文に対する生成コストを軽減することを提案する．

SuperSQL system refers to the database dynamically and can generate HTML documents by using the INVOKE function. Before we proposed caching system for SuperSQL which caches HTML documents generated dynamically by INVOKE function. The system has decreased the cost of regeneration for the same request. However, when the result that only the layout is different to the identical data was necessary, the cache could not be applied. In this paper, we propose the method of cashing the immediate data before changing into HTML documents for INVOKE function, and use the suitable cache for similar request by comparing the data structure of SuperSQL query.

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ページを動的に作成して提供する手法がある．慶應義塾大学遠山研究室で開発しているSuperSQL[1,2]では，このような動的にWebページを生成するための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，リンク先で評価すべきSuperSQL質問文および検索条件が埋め込まれており，リンクをクリックするたびにCGIを用いてSuperSQL処理系を起動し，パラメータによって動的にHTML文書を生成してユーザに提示を行う．

文献[3]では，このINVOKE関数によって動的に生成されたHTML文書をキャッシュすることで，同一リンクに対する複数回のアクセスに対して，同一のHTML文書を迅速に返すことができるようになった．しかし，これが利用できるのは，呼び出されたSuperSQL質問文が検索条件も含めて同一である場合のみであり，例えばブラウザの大きさによってレイアウトを変えるようなACTIVIEW[4]などでは，これらのキャッシュを利用できなかった．

そこで本論文では，HTML文書に変換する前段階のデータをキャッシュすることによって，INVOKE関数で指示されるSuperSQL質問文の構造を，キャッシュ生成時と比較することにより，適切なキャッシュを用いて類似質問文に対する生成コストを軽減し，ユーザへの応答時間を短縮することを目的とする．

2. SuperSQL と INVOKE 関数

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，指定された対象メディアへの出力を行う処理系である．SuperSQLでは特にSQLのターゲットリストを拡張した構文TFEを用い，データのレイアウトを規定する．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，反復子は，反復させたい部分を大括弧「[ ]」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，括弧の外側にある属性をキーとしてグルーピングすることができる．

特に，HTML文書をターゲットとする場合には，結合子の「%」はリンクによる結合を表し，前の属性の文字列に後に続くTFEに対するリンクを生成する．例えば，

title % [*actor*]!

といったTFEの場合は，*title*の文字列にリンクが生成され，そのリンク先にその*title*をキーとしてグルーピングされた*actor*の一覧の表を生成する．

2.2 INVOKE 関数

SuperSQLではINVOKE関数を用いることで，複数のSuperSQL質問文の間に，リンクによるナビゲーション機能を実現することができる．INVOKE関数は，呼び出すべき質問文のファイル名や属性値をパラメータに埋め込んだリンクを生成し，このリンクをたどることで，SuperSQLを呼び出し動的に生成することが可能である．INVOKE関数の書式は以下の通りである．

att % INVOKE(*queryfile,selection_cond,selection_att*)

引数には，評価するSuperSQL質問文のファイル名を示す*queryfile*と，その質問文に付加する検索条件に用いるための条件文*selection_cond*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_att*を指定する．また，INVOKE関数の前に 「%」を用いてリンクされる文字列を表す属性（*att*）が必要である．

このINVOKE関数を用いた動的なHTML文書を生成する処理の流れについて，図1に示す．まずINVOKE関数を含むSuperSQL質問文からHTMLを生成すると，「%」の直前の属性 *att* を評価した文字列に対して，SuperSQL の CGI である *ssql.cgi* へのリンクを生成する．このリンクをユーザがクリックすると， *queryfile* の質問文に *selection_cond, selection_att* で指定された条件を付加し，Web サーバ内の

♥ 学生会員 慶應義塾大学大学院理工学研究科後期博士課程 ari@db.ics.keio.ac.jp
♦ 学生会員 慶應義塾大学大学院理工学研究科修士課程 kyoko@db.ics.keio.ac.jp
♠ 正会員 慶應義塾大学理工学部情報科学科 toyama@ics.keio.ac.jp

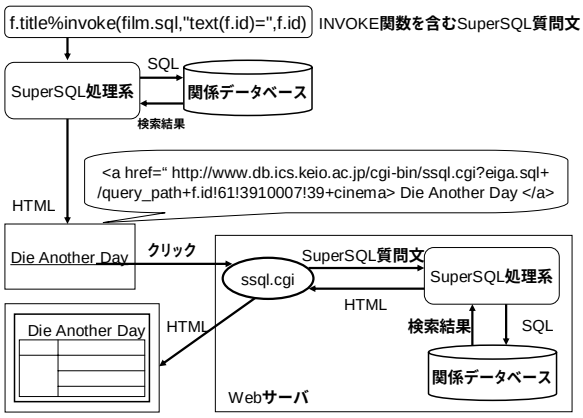


図 1 INVOKE 関数を用いた動的な HTML 文書の生成
Fig.1 Generating HTML documents using INVOKE function

SuperSQL 処理系に問い合わせ，結果をユーザに提示する．

2.3 生成結果のキャッシング

INVOKE関数によるリンクで実行されるCGIプログラム *ssql.cgi*は，ユーザからの要求ごとSuperSQL処理系を呼び出していた．ここで，処理されるSuperSQL質問文が複雑な場合は，データベースへのアクセスやデータの構造化にかかる時間が長く，HTML文書を出力するまでのユーザへの応答時間が長くなってしまう．特にアクセスが集中すると，データベースへのアクセスの際に大きな遅延が生じてしまう．

そこで，従来の研究[3]では，必要とするSuperSQL質問文内で利用されている表のタプルが更新されていないならば，同一のSuperSQLで生成されるHTML文書は同一になることに着目し，SuperSQL処理系を呼び出しているWebサーバのCGIプログラム側で，SuperSQL処理系で動的に生成されたHTML文書をキャッシュする手法を提案した．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生成結果から，ユーザに迅速にHTML文書を提示することが可能になった．

2.4 キャッシュ対象の拡張

SuperSQL処理系では図2のように，以下の3つの操作を経てデータベースの検索結果からHTML文書を生成している．

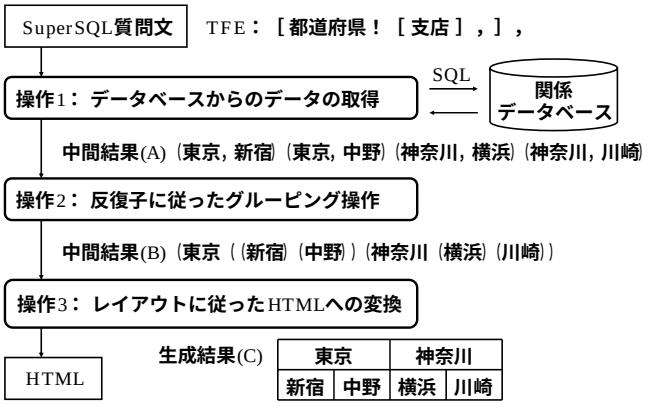


図 2 SuperSQL 処理系における HTML 文書の生成過程
Fig.2 Process of generating HTML documents on SuperSQL system

まず操作1では，SQLを用いて関係データベースからフラットなデータの取得を行う．データベースへのアクセスが必要なため，発行されるSQLによっては多くの時間を要する．特にSuperSQL質問文では，生成結果のグルーピングを生かすことのできる1対多や多対多の関連の結合を利用することが多いことから，データベースへの問い合わせコストが大きくなってしまう可能性がある．

この結果得られたフラットなデータは，操作2で反復子に従ってグルーピング操作を行い木構造データに変換を行う．この操作によって，フラットな表に存在する繰り返しを含むような中間結果を，グルーピングに従って小さくすることができる．この操作2では，グルーピングを行うためにフラットなデータに対して部分的ソートを行う必要がある．ソート操作の一部は，SQLでアクセスする際にORDER BY節を適切に与えること[5]で代替が可能であるが，複数のグルーピングが並列している場合では，ORDER BY節のみではグルーピングに必要なソートを全て行うことはできない．

この木構造データに対して，操作3でSuperSQL質問文のレイアウトに従って変換を行い，HTML文書を生成する．

従来の研究では，操作3の結果のHTML文書(図2の生成結果(C)にあたる)のみをキャッシュしていた．しかし，レイアウトだけが異なるSuperSQL質問文の場合は，操作1,2までは共通であるにもかかわらず，キャッシュを利用することができなかった．例えば，図2の例で「支店」を並べる方向を縦方向にする場合，必要となるデータは一緒であるにもかかわらず，最終的なHTML文書が異なるため，従来のHTML文書のキャッシュ手法ではキャッシュを利用できず，SuperSQL処理系を呼び出しデータベースへのアクセスから行う必要があった．

そこで，本論文では操作1の後の中間結果(A)を「DBスナップショット」，操作2の後の中間結果(B)を「木構造データ」と呼ぶこととする．そして，一度生成されたこれらの中間結果をWebサーバ側でキャッシュすることで，同一ではないが類似のSuperSQL質問文に対してDBスナップショットや木構造データのキャッシュからHTML文書を生成することによって，INVOKE関数が設定するリンクによるHTML文書の動的生成に対する応答時間を短縮することを目指す．以下では，DBスナップショットおよび木構造データについて，キャッシュの適用例を示し，キャッシュすることの有効性およびキャッシュ構築の上で必要なメタデータについて述べる．

3. DB スナップショットのキャッシュ

DBスナップショットのキャッシュは，SuperSQL処理系が発行したSQLをデータベース問い合わせた結果を，そのままの形でWebサーバに保存する手法である．

3.1 キャッシュの適用例

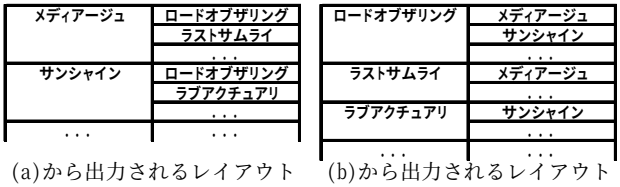
例えば，映画館とその上映タイトルという関連をあらわす表があり，この表に対して次のような二つのSuperSQL質問文があるとする．

(a) GENERATE html [映画館 ,[タイトル]!]!
FROM 映画館データ

(b) GENERATE html [タイトル ,[映画館]!]!
FROM 映画館データ

(a)は「映画館ごとに上映タイトルを並べる」という質問文，(b)は「タイトルごとに上映映画館を見たい」という質問文である．それぞれの生成結果のレイアウトを図3に示す．

この二つのSuperSQL質問文に対して，SuperSQL処理系では，図2の操作1にあたる関係データベースへの問い合わせ



(a)から出力されるレイアウト (b)から出力されるレイアウト

図3 質問文(a),(b)で生成するHTML文書のレイアウト
Fig.3 HTML documents generated from queries (a) and (b)

として，それぞれ以下に示すSQLが発行される．

(a') SELECT 映画館, タイトル FROM 映画館データ
(b') SELECT タイトル, 映画館 FROM 映画館データ

この(a')と(b')のSQLは，属性の射影の順序が異なるだけであり，この結果生じるビュー，つまり図2の中間結果Aは全く等価な情報を持っている．言い換えれば，(a')の結果から(b)のHTML文書を生成可能であり，(b')の結果から(a)のHTML文書を生成可能であるということである．

そこで，(a)によってHTML文書を動的に生成する過程で，図2の中間結果(A)にあたる(映画館，タイトル)を属性にもつビューをDBスナップショットとしてキャッシュしておくこととする．後に(b)によってHTML文書を動的に生成する場合，SQLの結果が変わらないことが保証されていれば，先にキャッシュした(a)のDBスナップショットを利用して，データベースへのアクセスなしにHTMLを生成することができる．

この例のように，対象となるビューが等しくなるようなSuperSQL質問文が複数ある場合に，DBスナップショットをキャッシュとして格納することで，二回目以降の生成に対してデータベースへのアクセスを省略することが可能である．

3.2 キャッシュの有効性

DBスナップショットのキャッシュでは，グルーピング操作を行う前のデータを保持しているため，グルーピング操作のキーが異なる場合でも，SQLが等価であれば利用することが可能になる．つまり，キャッシュの適用範囲は広いといえる．特に3.1節の例のように一つのビューに対してさまざまな面から見る場合には，このキャッシュは有効である．また，データベースへ問い合わせるSQLが複数の表を結合するなど計算量が大きくなる場合は，データベースへのアクセスを省略できるため，大きな効果が期待できる．

しかしながら，DBスナップショットはビューをそのまま表形式で格納する必要があるため，非常に大きな格納領域が必要になる可能性がある．反復されるグルーピングのキー項目を一つにまとめているのは，操作2のグルーピング操作であり，キャッシュの時点ではその恩恵を得ることができない．

3.3 キャッシュのメタデータ

DBスナップショットでは，SuperSQL質問文から発行されたSQLが等価であり，そのSQLが参照する表が更新されていないことが，キャッシュを利用するための条件となる．

したがって，DBスナップショットでは，中間結果とともにそのデータを生成したSQLの情報をメタデータとして格納し，比較に用いることが必要となる．具体的には，SuperSQL質問文から発行されたSQLに関する以下の情報を，メタデータとして保持する必要がある．

- SELECT節に書かれている属性集合のリスト
- From節に書かれている表名リスト
- Where節以下の条件

また，データベースアクセスの最適化として，一つのSuperSQL質問文に対して複数のSQLが発行されるときがあるが，この場合はそれぞれのSQL文に対してキャッシュを格納し利用することになる．

4. 木構造データのキャッシュ

木構造データのキャッシュは，DBスナップショットに対して，SuperSQL質問文内の反復子で指定されるグルーピングを行った結果として得られた木構造をもったデータに対し，キャッシュを行いWebサーバに保存する手法である．

4.1 キャッシュの適用例

再び3.1節の映画館データベースの例を用いる．この表に対して次のような二つのSuperSQL質問文があるとする．

(a) GENERATE html [映画館 ,[タイトル]!]!
FROM 映画館データ
(c) GENERATE html [映画館 ![タイトル]!],
FROM 映画館データ

(a)は3.1節でも用いた質問文であり,(c)は(a)に対して結合子と反復子を変更したものである．(c)の生成結果の例を図4に示すが，(a)の生成結果の例である図3と比較してもわかる通り，これら二つの質問文は，データのレイアウトする方向だけが異なり，HTMLを構成するために必要なデータは全く同じ質問文である．

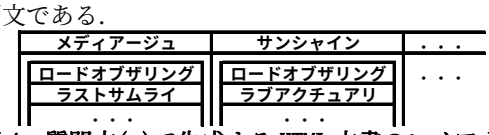


図4 質問文(c)で生成するHTML文書のレイアウト
Fig.4 HTML document generated from query (c)

まず,データベースへの問い合わせを考えると，(c)の質問文に対してSuperSQL処理系は次の(c')のSQLによってデータベースへの問い合わせを行うことになる．

(c') SELECT 映画館, タイトル FROM 映画館データ

これは(a')と全く同じSQLであり，この時点でDBスナップショットのキャッシュが利用できることがわかる．

しかし，(a)と(c)のHTML文書の生成の過程を比較すると，3.1節の(a)と(b)の質問文の関係とは異なり，DBスナップショットを生成した後，その次の木構造データの生成まで同一操作を行っている．言い替えると，データベースからのデータを取得後には，図2の操作2に示すように，反復子に従ったグルーピング操作を行うが，質問文(a)と(c)ではどちらも「映画館ごとに上映タイトルをグルーピングする」といった操作が必要になり，中間結果(A)が等しい状況下では，操作2によって生成された木構造データである中間結果Bは等しくなるということである．したがって，(a)を生成する時に生成される木構造データから(c)のHTML文書を生成することが可能であり，逆に，(c)を生成する時に生成される木構造データから(a)のHTML文書を生成することが可能であるということである．

この操作2にあたるグルーピング操作では，グルーピングのキー項目でのソートが必要である．そのため，対象となるタプルが多い場合や，再帰的にグルーピングを行っている場合では，グルーピングに非常に多くの時間を必要としている．そこで，(a)による動的なHTML文書生成が行われる過程で，図2の中間結果Bにあたる,映画館ごとに上映タイトルをグルーピングしたデータを木構造データとしてキャッシュすると，後に(c)によってHTML文書を動的に生成する場合，SQLの結果が変わらないことが保証されていれば，先にキャッシ

ュした(a)の木構造データを利用して，データベースへのアクセスを行わず，またグルーピング操作をしないことで，より高速にHTMLを生成することが可能となる．

この例のように，対象をなるビューが等しく，属性間のグルーピングの関係が等しいSuperSQL質問文が複数ある場合に，木構造データをキャッシュとして格納することで，2回目以降のHTML文書の生成の要求に対して，データベースへのアクセスおよびデータのグルーピング操作を省略することが可能となる．

4.2 キャッシュの有効性

木構造データは，グルーピング操作まで終わっているため，中間結果の大きさがDBスナップショットと比較して小さくできる．また，最終結果となるHTML文書と比較しても，HTMLタグでの装飾がない分だけ小さな領域しか用いないで済む．さらに木構造データは，結合子の結合方向や，反復子の反復方向，そして装飾指定などのレイアウトに全く依存しないため，HTML文書でキャッシュする場合より多くのSuperSQL質問文がこのキャッシュを利用できる．

しかし，グルーピング操作によって，DBスナップショットの時点では保持しているSQLの結果のタプルとしての属性値間の関連の情報が一部欠落してしまう．したがって，等価なSQLを利用していても，別の項目をグルーピングのキーとするSuperSQL質問文に対しては，DBスナップショットとは異なり，この木構造データのキャッシュを利用できない．

4.3 キャッシュのメタデータ

木構造データのキャッシュを利用するには，DBスナップショットのキャッシュの利用条件に加えて，属性間のグルーピングの関係がキャッシュ取得時と同等である必要がある．この属性間のグルーピングの関係を表す方法として，グルーピング木[5]がある．グルーピング木はどの属性がどの属性によってグルーピングされているかを，木構造の親子関係を用いて表したものであり．このグルーピング木が等しければ木構造データのキャッシュが適用可能であるといえる．

例えば，これまでに例に挙げてきたSuperSQL質問文(a),(b),(c)に対するグルーピング木は図5のようになる．

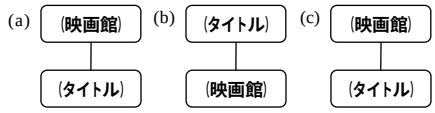


図5 質問文(a),(b),(c)に対応するグルーピング木
Fig.5 “Grouping Trees” for queries (a),(b) and (c)

図5から，(a)と(c)のグルーピング木は同じであり，(b)のグルーピング木はそれらとは異なるため，(a)と(c)は木構造データのキャッシュを相互に利用できるが，(b)とは木構造データのキャッシュを相互に利用できないと判断できる．

したがって，木構造データのキャッシュでは，キャッシュのメタデータとして，DBスナップショットにおけるメタデータに加えグルーピング木を同時に格納することが必要である.後にINVOKE関数によってSuperSQL質問文が与えられたときには，そのSuperSQL質問文に対するグルーピング木と比較することで，木構造データのキャッシュを適用できるか否かを判断する．

5. キャッシュの性能比較

これまで述べてきたDBスナップショット，木構造データのキャッシュと，従来のHTML文書のキャッシュの性能の比較をまとめると，表1のようになる．

	DBスナップショット	木構造データ	HTML文書
キャッシュの汎用度	○	△	×
キャッシュ一つあたりのデータの大きさ	××	○	△
キャッシュからのHTML文書生成時間	×	△	○

表1 3種類のキャッシュの性能比較
Table 1 Performance comparison with three caches

6. おわりに

本論文では，SuperSQL処理系におけるINVOKE関数による動的なHTML文書の生成において，DBスナップショットおよび木構造データをキャッシュする手法について述べた．これにより，従来のHTML文書のキャッシュだけでは利用できなかった類似のSuperSQL質問文に対して，キャッシュを利用することが可能になり，動的なHTML文書生成時にWebサーバの応答時間を短縮することができると考えられる．

DBスナップショットと木構造データ，HTML文書のキャッシュ手法は，それぞれ，キャッシュの汎用性，大きさ，キャッシュからのHTML文書生成時間に一長一短の特性を持つ．このため，実際に処理される問い合わせ状況を考慮したキャッシュの設計を行う必要があると考えられる．

[文献]

[1] 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).
[2] SuperSQL HOME PAGE, http://www.db.ics.keio.ac.jp/ssql/index.html
[3] 有澤達也,石川恭子,遠山元道: SuperSQL 処理系における INVOKE 関数に対するキャッシュ機構, 情報処理学会研究報告, DBS-131-037, (2003).
[4] Yoko Maeda, Motomichi Toyama: ACTIVIEW: Adaptive data presentation using SuperSQL, VLDB 2001, pp. 695 - 696, (2001).
[5] 有澤達也,遠山元道: SuperSQL 処理系におけるグルーピング操作の効率的な実装,データ工学ワークショップ (DEWS2001), (2001).

有澤 達也 Tatsuya ARISAWA
慶應義塾大学大学院理工学研究科後期博士課程在学中. 1999 慶應義塾大学大学院管理工学科修士課程修了．データベースシステムの研究に従事．情報処理学会学生会員．日本データベース学会学生会員.

石川 恭子 Kyoko ISHIKAWA
慶應義塾大学大学院理工学研究科修士課程在学中．2003 慶應義塾大学理工学部情報工学科卒業．データベースシステムの研究に従事．情報処理学会学生会員．日本データベース学会学生会員.

遠山 元道 Motomichi TOYAMA
慶應義塾大学理工学部情報工学科専任講師．博士（工学）．1984 慶應義塾大学大学院博士課程単位取得退学．主にデータベースシステムの研究に従事．IEEE Computer Society, ACM, 情報処理学会，日本ソフトウェア科学会，電子情報通

信学会，日本データベース学会会員.