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

2007－DBS－143 (87)
2007／7／4

SuperSQLを利用した複雑な集約を可能とするデータキューブ機構の提案

中道 亮† 遠山 元道††

† 慶應義塾大学大学院 理工学研究科 開放環境科学専攻 〒223-8522 横浜市港北区日吉 3-14-1
†† 慶應義塾大学 理工学部 情報工学科 〒223-8522 横浜市港北区日吉 3-14-1
E-mail: †ryo@db.ics.keio.ac.jp, ††toyama@ics.keio.ac.jp

あらまし 本稿ではSuperSQLを用いて複雑な集約を可能とするデータキューブ機構を提案する。データ分析において重要なデータだけを格納したデータキューブを作成することはデータ分析の効率を向上させる為に重要であると言えるが、データキューブを構築する際に用いるCUBE演算子は全ての次元の組み合わせの集約値からなるクロス集計値を2次元のフラットな表として返すCUBE演算子ではユーザの複雑な要求を必ずしも満たすことは出来ない。そこで様々な次元での集約値を保持出来るSuperSQL処理系を用い、異なるレベルでの集約値比較によるフィルタリングや、フィルタリングしたグループに対し再度集約演算を行うといった追加問合せ可能とすることで各ユーザの細かい要求を実現する。また、問合せ結果をWeb上に出力し可視化することで操作性の向上を目指す。
キーワード データキューブ、DB言語、SuperSQL

A Proposal of DataCube which support Complex Aggregation by using SuperSQL

Ryo NAKAMICHI† and Motomichi TOYAMA††

† School of Science for OPEN and Environmental Systems,
Faculty of Science and Technology, Keio University
Hiyoshi3-14-1, Kouhoku-ku, Yokohama-shi, 223-8522 Japan
†† Department of Information and Computer Science, Faculty of Science and Technology,
Keio University Hiyoshi3-14-1, Kouhoku-ku, Yokohama-shi, 223-8522 Japan
E-mail: †ryo@db.ics.keio.ac.jp, ††toyama@ics.keio.ac.jp

Abstract This paper propose the way to develop datacube which support complex aggregation by using SuperSQL. At data analysis, making a datacube with only important datas will make data analysis efficiently. However cube operation will calculate aggregate values by using values calculated at all dimention, and will return 2-D table. Because of this process, cube operation sometimes could not satisfy user's request. So by using SuperSQL which can hold all aggregate values at any dimention, we will propose datacube which support additional querying such us filtering at different dimention and additional querying to filtered group to satisfy user's request. In addition, we will show results to the web to improve user-friendliness.
Key words Data Cube, DB Language, SuperSQL

1. はじめに

企業のデータウェアハウスに格納されているビジネスデータに対する多角的な分析業務を効果的にサポートする手法としてOLAPと呼ばれる多次元データモデルに基いた分析手法がある。OLAPとは複数の次元から成り立つデータキューブ[1]を作成してキューブ内のセルに変数を格納し、様々な切り口からデータの集合体を眺めることで、全体の傾向や問題点などを見つけだすというものである。データキューブでは指定された次元属性の全ての組み合わせでデータに対しグルーピングを行い、それぞれの組み合わせの集約値を全て格納する。しかし、データ分析において分析の効率を上げる為、各ユーザの細かなニー

ズに応え重要な集約値だけをデータキューブに格納できることが求められる。その際に重要となってくる機能として非集約値と集約値との比較及び1つのレベルの集約値と異なるレベルの集約値との比較を行うことによってデータのフィルタリングを行い、フィルタリングされたデータに対し再度集約演算処理を行うというような追加問い合わせ機能が挙げられる。しかし、現状では追加問い合わせを施したデータキューブを生成することは不可能である。そこで本研究では異なるレベルでの様々な集約値を保持できるSuperSQL処理系[4][5]を用い、ユーザの複雑な集約演算要求を満たすデータキューブ機構を実現する。以下、本稿の構成を示す。まず、2.章でSuperSQL処理系について述べる。次に3.章ではデータキューブに関する説明及び問題点について説明し、4.章で本研究における提案を説明する。そして、5.章において関連研究との比較を行い、6.章において今後の課題および結論を述べる。

2. SuperSQLとは

SuperSQLはSQLを拡張したワンソースマルチユースを実現する言語である[4][5]。その質問文は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。ここで < *media* > は出力媒体を示し、HTML、XML、Excel、LaTeX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。括弧内は左がクエリー中の演算子を示し、右がレイアウト式を示す。レイアウト式については??節で説明する。

- 水平結合子(, : C1)

データを横に結合して出力。

例: Name, Tel

name	tel

- 垂直結合子(! : C2)

データを縦に結合して出力。

例: Name! Tel

name
tel

- 深度結合子(% : C3)

データを3次元方法へ結合。出力がHTMLならばリンクとなる。

例: Name % Tel

name → tel

また時間軸方向結合として、結合子「#」も存在する[?]。

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かえして表示する。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 , 評点]!]!

とした場合には、その科目において学生の評点一覧が表示されるが、

[科目名]! , [学籍番号]! , [評点]!

とした場合には前者のような関連はなく、単に各々の一覧が表示されるだけである。以下その種類について説明する。

- 水平反復子([ ], : G1)

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す。

例: [Name],

name1	name2	···	name10

- 垂直反復子([ ]! : G2)

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す。

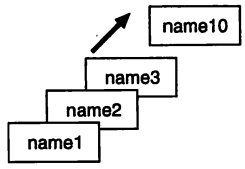
例: [Name]!

name1
name2
···
name10

- 深度反復子([ ]% : G3)

データインスタンスがある限り、その属性のデータを奥行き方向に繰り返す。

例: [Name]%



2.3 集約演算

異なるレベルにおいて様々な集約演算を行うことが可能である。SuperSQL処理系では標準の集約関数として最大値(max)、最小値(min)、平均(avg)、合計(sum)、頻度数(count)が用意されている。標準的なSQLと同様に演算対象属性を括弧で括り、その括弧の前に適用する集約演算を記述する。例えば

[名前]!, avg(評点)

の様に記述することで、学生名及びや学生の平均評点を得ることが可能である。また、avgの代わりにmaxを書くと評点の中で一番高い値、minを書くと一番低い値を求めることが可能である。

また異なるレベルにおいて異なる集約関数を複数指定することも可能である。例えば

```
SELECT 月, 都市, 商品, sum(売上数)
FROM 売上
GROUP BY CUBE 月, 都市, 商品
```

図 1　売上キューブを生成するクエリ

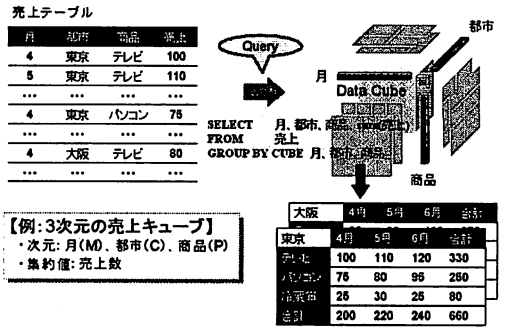


図 2　データキューブ生成の流れ

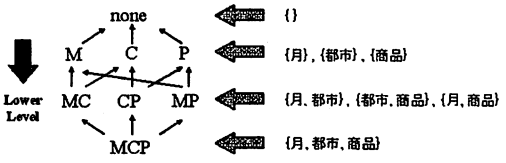


図 3　売上キューブの構造

[[科目, sum(評点)]! avg(評点)]!

の様に記述すると、科目別の評定の合計点及び全体の平均評点を求めることが可能である。

3. データキューブ

3.1 データキューブ生成方法

データキューブ [1] は CUBE 演算子と呼ばれる演算子によって生成することが出来る。CUBE 演算子は指定された次元全ての組み合わせでデータの集約を行い集約値を計算し、その結果全てを 2 次元のフラットな表にまとめてユーザへ返す。以下月、都市、商品、売上数という属性を持つ売上テーブルから 3 次元の売上キューブを生成する例を説明する。売上テーブルから月 (M)、都市 (C)、商品 (P)、集約値を売上数とした 3 次元の売上キューブを生成するには図 1 の様なクエリを記述する。GROUP BY 句に CUBE 演算子を記述し、次元として月、都市、商品を指定することで CUBE 演算子は指定された次元全ての組み合わせでデータの集約を行い集約値を計算する。この際、売上キューブは図 3 の様な構造を持つことになる。図 2 に月 (M)、都市 (C)、商品 (P)、集約値を売上数とした 3 次元の売上キューブを生成する流れを。注意点として図 2 中ではキューブ生成結果をクロス集計表として表しているが、図 2 は概念図として分かりやすく表しており、実際の生成結果は 2 次元のフラットな表で返される。

3.2 問題点

CUBE 演算子では各次元の組み合わせによる単純なクロス集計を行うのみであり、ユーザの複雑な集約要求を満たせるとは限らない。例えば図 4 に示す様な「都市別で各商品における年間平均売上数よりも月の売上数が少なかった月数」という複雑な集約例の演算結果を得ることは CUBE 演算子では不可能である。以下の章では上記の例を用いて CUBE 演算子の問題点を挙げる。

3.2.1 集約演算指定

CUBE 演算子では各レベルにおいて異なる集約演算を指定することが出来ない。例えば図 2 の売上キューブにおいて MCP では合計を計算し、その一段階上のレベル (MC, CP, MP) ではその平均値を計算するといったような異なるレベルで異なる集約演算処理を行うことは不可能であり、上記の例での MCP で月の売上数である *sum*、CP では年間平均売上である *avg* を計算するといった部分が不可能であるということを示している。

3.2.2 HAVING 句の機能

集約演算結果を用いてフィルタリング処理を行うことは HAVING 句を用いることによって実現可能である。しかし、HAVING 句では全てのレベルの集約演算結果に対して同じ条件しか付与することが出来ない為、多段階のレベルを持ちレベルによって集約値が大きく異なるデータキューブにおいては必ずしも有効であるとは言いがたい。また、HAVING 句を用いて集約値に対して条件を付与することは可能であるが、異なるレベルの集約値同士を比較するような記述を行うことは不可能である。その為、異なるレベルでの集約値を比較して何らかの処理を行うということは不可能である。例えば MCP における平均値と CP における平均値とを比較するといったことは不可能であり、上記の例での年間平均売上数よりも月の売上数が少なかった月を求める部分が不可能であるということを示している。

3.2.3 追加集約演算

上位レベルの集約値を用いて下位レベルのグループに対しフィルタリングを行い重要なグループだけを残し、再度そのグループに対し集約演算を行うといった処理は複雑な集約要求を満たす為には非常に有効であると言えるが、現状ではその様な処理を行うことは不可能である。上記の例での年間平均売上数よりも月の売上数が少なかった月を求め、再度その月数を求める *count* を計算するという部分が不可能であるということを示している。

3.2.4 断面別条件指定

上記の例はデータキューブの特定の断面 (月、商品) のみに特化した要求であり、そのフィルタリング条件の元でデータキューブが生成されるが、それぞれの断面における 2 次元の組み合わせによる小計と 3 次元の組み合わせによる小計を比較したいといった要求も考えられる。しかし、CUBE 演算子では (月、商品) 断面では MCP を CP と比較し、(都市、商品) 断面では MCP を MP と比較するといったような処理は行うことが出来ない。

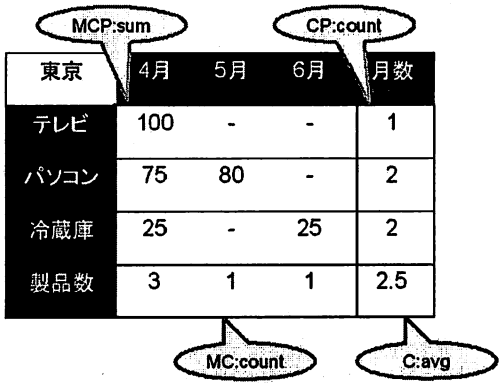


図 4　CUBE 演算子では取得不可能な結果例

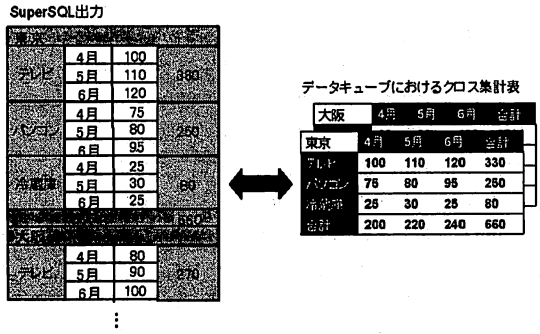


図 5　SuperSQL 処理系における出力例

3.3　SuperSQL 処理系の出力とデータキューブにおけるクロス表の比較

SuperSQL 処理系を用いた多段階の集約を施した出力例を図 5 に示す。図 5 内を見ても分かるように、異なるレベルにおける集約値を保持することが可能ではあるが、SuperSQL 処理系では集約順序を決定し集約を行う為、クロス集計表内における MC の集約値が取得できないなど、上位レベルにおいて取得できない集約値が存在する。しかし、最下層の MCP の集約値であればスライシング、ダイシングといった基本機能を備えたデータキューブと同等の情報を保持することが出来る。

4.　提案手法

遠山研究室で開発されている SuperSQL 処理系では問い合わせ結果を容易に構造化することが可能であり、各レベルで異なる集約演算の結果を問い合わせ結果中に保持出来るという利点がある。また、著者らによって SuperSQL 処理系においてプラグイン関数機能が実現されている [6]。プラグイン関数を用いることによってユーザはユーザ定義関数を定義することが可能なので標準的な集約関数を用いるだけでは不可能である複雑なユーザの集約演算及び出力結果の要求を満たすことが可能である。また SuperSQL 処理系では、あるレベルの集約演算結果をクエリ内でローカル変数として保持し、異なるレベルでの参照を可能とする assign 関数 [6] と呼ばれる関数を使用することが出来る。この関数を利用することで異なるレベル間での集約値比較が可能であると考え、本研究では SuperSQL 処理系を拡張することで異なるレベル間での集約値比較によるグループのフィルタリング機能を実現し、更に追加問い合わせ機能を実現することで複雑な集約を可能とするデータキューブ機構を実現する。

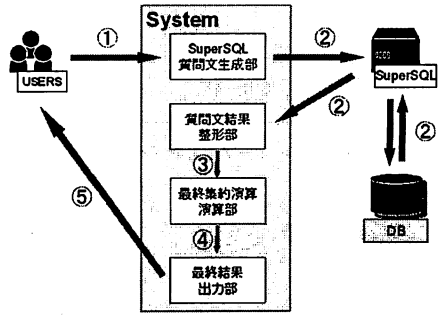


図 6　システム概要

4.1　提案システム

図 6 が提案システムの概要である。追加問い合わせで重要となる機能として「集約値比較によるデータのフィルタリング」と「フィルタリング後のデータに対する集約演算」がある。3.3 章及び 4. で述べた理由より本研究では、前者の機能は SuperSQL 処理系に HAVING 句を導入し、assign 関数を利用することによって実現し、後者の機能はシステム側で実現している。ユーザが本研究における提案システムを利用する際に必要となるものは、データキューブ生成に必要な元となるデータを取得する為の簡単な SQL 質問文とシステムとのインタラクション用 HTML フォームのみである。ユーザからシステムに与える情報は「選択次元及び集約値」、「フィルタリング条件」、「中間結果及び最終結果に用いる集約演算」である。

4.2　HAVING 句の導入

本研究ではフィルタリング機能を実現する為に SuperSQL 処理系へ HAVING 句を導入した。HAVING 句には assign 関数を用いて各レベルの集約値をローカル変数として保持し、ローカル変数名を直接記述する。図 7 が HAVING 句を用いて記述した SuperSQL クエリ例である。以下クエリについて説明する。このクエリは都市、商品、月の順で集約を行う。4 行目の *sum* で都市、商品、月別の売上合計数を求めている。これはクロス集計表内の MCP の集約値に相当する。5 行目の *avg* で都市、商品別の平均売上数を求めている。これはクロス集計表内の CP の集約値に相当する。また assign 関数によって第 1 引数の結果が第 2 引数で指定された名前を持つローカル変数として保持されている (4 行目では各グループの *sum*(*s.sales*) の結果が mcpsum という変数名で保持され、5 行目では各グループの *avg*(*s.sales*) の結果が cpavg という変数名で保持される)。そして 9 行目の HAVING 句における *mcpsum* < *cpavg* という条件によって、「都市別で各商品における年間平均売上数よりも月の売上数が少なかった月」という要求を満たす結果を得ることが出来る。図 7 のクエリにおいて HAVING 句を記述した場合と記述しなかった場合の実行結果を図 8 に示す。

```
1  GENERATE HTML [
2    s.city ,
3      [ s.product ,
4         [ s.month, assign(sum(s.sales), mcpsum) ]! ,
5        assign(avg(s.sales, cpavg)
6      ]! ,
7  ]!
8  FROM sale_info s
9  HAVING mcpsum < cpavg
```

図 7　HAVING 句を用いた SuperSQL クエリ例

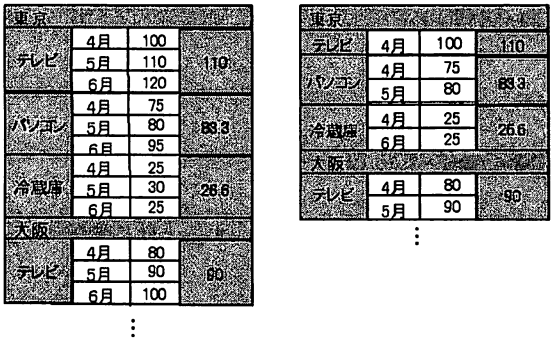


図 8　(左)HAVING 句無し (右)HAVING 句あり

4.3　キューブ生成アルゴリズム

ユーザから指定される条件としては、

（1）　各断面毎に条件を指定

（2）　単一条件のみを指定

の二通りが考えられる。本研究では 4 つのフェーズからなるアルゴリズムによってキューブを生成する。以下の章でキューブ生成アルゴリズムを説明する。

4.3.1　SuperSQL クエリ生成フェーズ

ユーザとのインタラクションで得る元に中間結果、つまりフィルタリング後のデータをデータを生成する SuperSQL クエリを自動的に生成する。この際、フィルタリング条件によって SuperSQL クエリ内における集約順序を決定する。フィルタリング条件は断面指定と次元の組合せによる条件句としてシステム側に受け渡される。(1) の場合では断面を指定し、断面に対して $MCP < CP$ の様な条件を渡すことによってキューブの元となる選択次元の数だけ SuperSQL クエリが生成される。また (2) の場合では、断面の指定はせず $MCP < CP$ という条件のみを渡し、SuperSQL クエリは 1 つだけ生成される。(1) と (2) で生成される SuperSQL クエリ数が違う理由は、(1) では中間結果が生成される際フィルタリングされるグループがそれぞれの断面によって異なる為である。

4.3.2　中間結果生成フェーズ

SuperSQL クエリを SuperSQL 処理系に発行し、生成された中間結果をシステム側で受け取る。この際、(1) の場合は選択次元数だけ中間結果を受け取り、(2) の場合は 1 つのみ受け取る。

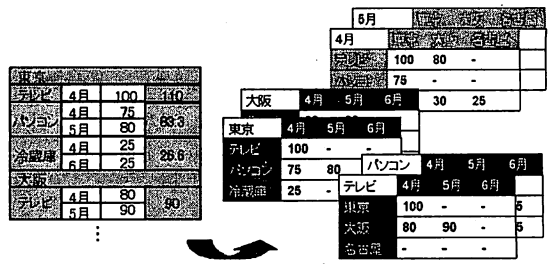


図 9　中間クロス集計表生成

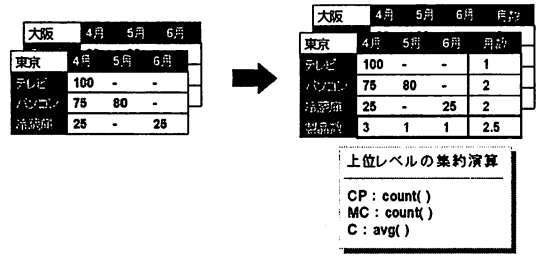


図 10　最終クロス集計表生成

4.3.3　中間クロス集計表作成フェーズ

SuperSQL 処理系から受け取った中間結果を元にスライシング、ダイシングを満たす全てのクロス集計表を作成する。(1) の場合であれば各中間結果を用いて各断面からみたクロス集計表を作成し、(2) の場合であれば受け取った 1 つの中間結果から全ての断面から見たクロス集計表を作成する。また、この際作成するクロス集計表には最下層レベルの集約値のみを記載する。図 9 に (2) の場合の処理イメージを示す。

4.3.4　最終クロス集計表作成フェーズ

中間クロス集計表とインタラクションで得た情報を元に上位レベルの集約演算を行いクロス集計表を完成させる。結果は Web 上に出力し、ユーザは出力された結果を用いて分析を行う。図 10 に (2) の場合の処理イメージを示す。

5.　関連研究と問題点

一度集約されたグループ内のタプルに対して集約値と比較して何らかの条件を付与しタプルのフィルタリングを行い、再度集約演算を行うといった様な複雑な集約演算を行うことは単一の SQL クエリのみで行うことは不可能であり、結果を求めるには必要なだけのビューを生成し、それらを結合することによってしか実現出来ない。そこで Damianos らは SQL を拡張しグルーピング変数と SUCH THAT 句という機能を持たせることによって単一の SQL クエリで実現し、ビュー生成の煩雑さ及び結合処理によって生じるコストを削減している [2]。また Ross らは CUBE 演算子でグルーピングされた各グループ内のタプルに

対しての追加問い合わせ実現した Multi-Feature Cube(MFC) と呼ばれるデータキューブ機構を提案している [3]。これは上記のグルーピング変数及び SUCHTHAT 句を用い、CUBE 演算子における次元の組み合わせ、つまり各レベルのグループにおいて同様の条件をグループ内のタプルに対して付与することが可能である。

5.1 既存研究の問題点

異なるレベルの集約値比較が追加問い合わせ機能では重要になってくるが、Ross らが提案した MFC では該当グループの集約値もしくは全体の集約値とグループ内のタプルを比較することによってタプルのフィルタリングを行うことは出来るが、集約値同士の比較を行うことは不可能でありグループに対するフィルタリングを行うことは出来ない。また CUBE 演算子自体の問題ではあるが、異なるレベルにおいて異なる集約演算を行うことは不可能であり、真の意味で複雑な集約を実現出来るとは言いがたい。本研究ではタプルと集約値との比較に加え、異なるレベルでの様々な集約値比較をきめ細やかに行いフィルタリングを行うことが可能である。

6. おわりに

本稿では、集約値比較によってグループのフィルタリングを行えるよう拡張した SuperSQL 処理系を利用することによって、ユーザの複雑な集約要求を可能としたデータキューブ機構を提案した。今後の課題については、データキューブにおける重要な機能のうち、スライシング及びダイシングについては本稿での提案で実現しているが、階層構造を持つ次元についてのドリルダウン機能については今のところ実現しておらず、今後検討していかなければならない。現在考えている実現方法としては、出力は Web 上で表示されるので、階層構造を持つ次元については値にリンクを張りクリックされると動的に SuperSQL クエリを生成し、その下位レベルのキューブを生成するという処理にするといった方法を考えている。また、ユーザの意図を適切にシステムに伝えるユーザインタラクションの方法についても更に考える必要があると考えている。

文　献

[1] J. Gray, A. Bosworth, A. Layman and H. Pirahesh. Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Total. In *Proceedings of the 12th International Conference on Data Engineering*, pp.152-159, 1996.

[2] D. Chatziantoniou and K. Ross. Querying Multiple Features of Groups in Relational Databases. In *Proceedings of the 22nd VLDB Conference*, pp.293-306, 1996.

[3] K. Ross, D. Srivastava and D. Chatziantoniou. Complex Aggregation at Multiple Granularities. In *Proceedings of the 6th International Conference on Extending Database Technology*, pp.261-277, 1998.

[4] M. Toyama. SuperSQL: An Extended SQL for Database Publishing and Presentation. In*Proceedings of ACM SIGMOD '98 International Conference on Management of Data*, pp. 584-586, 1998

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

[6] 中道兗, 金翰基, 遠山元道. SuperSQL 処理系におけるプラグイン関数および局所的な整列の実現. In 日本データベース学会 *Letters Vol.5, No.1*, pp. 65-68, 2006