2003－DBS－131　(10)
2003／7／16

SuperSQL クエリ作成支援系における情報容量の提示

大澤 幸子† 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
‡慶應義塾大学 理工学部 情報工学科
E-mail: † sachi@db.ics.keio.ac.jp, ‡ toyama@ics.keio.ac.jp

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指定することを可能にした SQL の拡張言語である。しかし、レイアウトによっては結果表示においてクエリの対象となるデータベース上の属性間の関連が正しく反映されず、情報が適切に表現されないことがある。本研究では情報容量の概念を定義し、クエリの情報容量とデータベースの情報容量の大小関係による結果表示の分類を行った。また、SuperSQL クエリ作成時にクエリおよび データベースの情報容量を算出し、求められた両情報容量を比較することによって表示異常の有無を判定する手法を提案し、実現した。

キーワード：情報容量 、 SuperSQL 、 適正表示 、 表示異常

Information Capacity Presentation in the SuperSQL Query Support System

Sachiko OSAWA † Motomichi TOYAMA ‡

†School of Science for OPEN and Environmental Systems,
Faculty of Science and Technology, Keio University.
‡Department of Information and Computer Science, Faculty of Science and Technology,
Keio University.
E-mail : †sachi@db.ics.keio.ac.jp ‡toyama@ics.keio.ac.jp

SuperSQL is the extended language of SQL which made it possible to structurize the output result from a relational database, and to specify various layout expression. however, some layout cause a output result that lost a part of information on the target database of the query, and it may not be expressed appropriately. Then, this research defines the concept of information capacity and the classification of the output result based on the size relation between the information capacity of a database and the information capacity of a query. Moreover, We proposed and realized the tequnique of judging the existence of the abnormalities in a result output by computing the information capacities of a query and a database at the time of SuperSQL query creation, and measuring both the calculated information capacity

keyword : Information Capacity , SuperSQL , Propare Indication , Abnormal Indication

1　背景

SQL の拡張言語である SuperSQL[1, 2] では、簡単なクエリによって多様な出力メディア、表示レイアウトを指定することが可能である。表示レイアウトに関しては、同じ属性集合に対して多様なレイアウトの指定が可能であるという点が、SuperSQL の大きな特長の 1 つでもある。一般的に、データベース[1] に対してクエリを実行した表示結果は、そのクエリの対象となるデータベースに含まれる情報を的確に反映していることが望ましいと考えられる。

しかし、レイアウトによってはクエリ実行後の表示結果においてクエリの対象となるデータベース上の属性間の関連が正しく反映されず、情報が適切に表現されないことがある [4]。ユーザが意図的にそのようなクエリ指定を行っている場合は問題ないが、無意識に指定したクエリによってデータベース上の情報と、表示結果における情報が一致しないという状況はは好ましくない。

近年 SuperSQL に関する研究が盛んに行われていることから、今後そのユーザ層が拡大し、ユーザのデータベースや SuperSQL に対する知識レベルも多様化していくことが考えられている。それによって特別な意図が無いにもかかわらず、表示異常をもたらすレイアウト指定を行ってしまうユーザが現れることが容易に予想できる。このような状況において、ユーザがレイアウト指定と表示異常の関連について細かい知識を持っていなくとも、意図しない表示異常を回避できるよう、クエリ作成時におけるユーザ支援が求められている。

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。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[3] を表し、連結子、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　連結子

連結子はデータベースから得られたデータをどの方向 (次元) に連結するかを指定する演算子であり、以下の 3 種類がある。括弧内がクエリ中の演算子を示す。

- 水平連結子 (,)
  データを横に連結して出力。
  例：Name, Tel

name	tel

- 垂直連結子 (!)
  データを縦に連結して出力。
  例：Name! Tel

name
tel

- 深度連結子 (%)
  データを 3 次元方向へ連結。出力が HTML ならばリンクとなる。
  例：Name % Tel

name	→	tel

また時間軸方向連結として、時間連結子 (#) も存在する [6]。

2.2　反復子

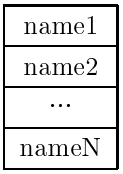
反復子は指定する方向に、データベースの値があるだけ繰りかえして表示する演算子であり、以下の 3 種類がある。括弧内はクエリ中の演算子を示す。

- 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す。
  例：[Name],

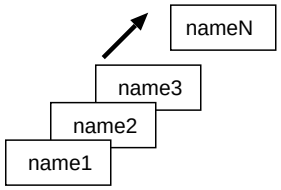
name1	name2	…	nameN

- 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す。
  例：[Name]!

[1] 但し、データベースは第 5 正規形であるとする



- 深度反復子 ([ ]%)
  データインスタンスがある限り、その属性のデータを奥行き方向に繰り返す。
  例 : [Name]%



また時間軸方向連結として、時間反復子 ([ ]#) も存在する [6]。

反復子の中に反復子を用いることで、SQL におけるグルーピングの効果を得ることができる。これにより、冗長な値の表示を制御し、より見やすい出力を得ることができる。つまり、反復子は単に構造を指定するだけでなく、そのネスト関係によって属性間の関連を指定することができるのである。

2.3 結果表示の例

結果表示においてクエリの対象となるデータベース上の情報が適切に表現されない状態について、具体的に例を挙げて説明する。クエリの対象とするテーブルを表 1,2 に示す。

表 1 のテーブル Train は、「どこの “都道府県” にどの “駅” がある」「どこの “ 都道府県” になんという “鉄道会社” が通っている」「どの “駅” にどの “鉄道会社” が通っている」という 3 組の属性間の関連を含んでいる。また表 2 のテーブル Hotel は、「どの “所在地” になんという “名称” のホテルがある」という 1 組の属性間の関連を含んでいる。ここで属性 “所在地” の値は、全てテーブル Train の属性 “駅” の値となっている。

ここで、表 1 に対しクエリ 1 を実行すると、結果は表 3 に示すようになる。

表 1: テーブル Train

都道府県	駅	鉄道会社
東京都	渋谷	JR
東京都	渋谷	都営地下鉄
東京都	渋谷	東京急行
東京都	品川	JR
東京都	品川	都営地下鉄
東京都	品川	京浜急行
神奈川県	横浜	JR
神奈川県	横浜	東京急行
神奈川県	横浜	京浜急行

表 2: テーブル Hotel

所在地	名称
渋谷	エクセルホテル東急
渋谷	東急イン
品川	品川プリンスホテル
品川	高輪京急ホテル
横浜	東急ホテル
横浜	横浜ベイシェラトン

クエリ 1 *GENERATE latex*
*[t.*都道府県*, [t.*駅*, [t.*鉄道会社 *]!]!]!*
FROM Train t

表 3 を見て分かるように、クエリ 1 は表 1 において繰り返し同じ値が現れるという冗長を排除し、意味的なまとまりを分かりやすく表現するものである。この表示結果は、クエリの対象となったデータベース、つまりテーブル Train が含む 3 組の属性間の関連を全て含み、「どこの “都道府県 ” の、どの “駅” には、どの “鉄道会社” が通っている」という情報が適切に表現 され、またそれ以外の過剰な情報は表現されていない。

次に、表 1 に対しクエリ 2 を実行した結果を表 4 に示す。

クエリ 2 *GENERATE latex*
*[t.*都道府県*, [t.*駅 *]!, [t.*鉄道会社 *]!]!*
FROM Train t

表 4 を見てわかる通り、クエリ 2 はそれぞれの都道府県に属する駅、鉄道会社を都道府県によって

表 3: クエリ 1 の結果 (適正表示)

東京都	渋谷	JR 都営地下鉄 東京急行
	品川	JR 京浜急行
神奈川県	横浜	JR 東京急行 京浜急行

表 4: クエリ 2 の結果 (過小表示)

東京都	渋谷 品川	JR 都営地下鉄 東京急行 京浜急行
神奈川県	横浜	JR 東京急行 京浜急行

それぞれグルーピングしている。そのため、 テーブル Train に含まれる 3 組の属性間の関連のうち、「どの “都道府県” にどの “駅” がある」 「どこの “都道府県” にどの “鉄道会社” が通っている」と いう 2 組の情報は表現されているが、 残りの「どの “駅” にどの “鉄道会社” が通っている」という情報が表現されていない。

最後に、表 1,2 に対しクエリ 3 を実行した結果を表 5 に示す。

クエリ 3 *GENERATE latex*
*[t.*駅*,[t.*鉄道会社*, h.*名称*]!]!*
FROM Train t, Hotel h
*Where t.*駅 *= h.*所在地

クエリ 3 はテーブル Train の “駅” とテーブル Hotel の “所在地” を結合条件としているため、表 5 ではそれぞれのテーブルに含まれる「どの “駅” にどの” 鉄道会社” が通っている」「どの “駅” に何という “名称” のホテルがある」という属性間の関連が表現されている。しかし一方で、クエリの対象となるデータベース、つまり上記 2 つのテーブルには含まれない「“鉄道会社” と “名称”」の間の関連が

表 5: クエリ 3 の結果 (過大表示)

渋谷	JR	エクセルホテル東急
	JR	東急イン
	都営地下鉄	エクセルホテル東急
	都営地下鉄	東急イン
	東京急行	エクセルホテル東急
	東京急行	東急イン
品川	JR	品川プリンスホテル
	JR	高輪京急ホテル
	京浜急行	品川プリンスホテル
	京浜急行	高輪京急ホテル
横浜	JR	東急ホテル
	JR	横浜ベイシェラトン
	東京急行	東急ホテル
	東京急行	横浜ベイシェラトン
	京浜急行	東急ホテル
	京浜急行	横浜ベイシェラトン

含まれてしまっているようにみえる。

このようにクエリの対象となるデータベースに含まれる情報が、表示結果において表現されない、また逆に、対象データベースには含まれない情報が存在するかのように表示結果に現れる、といった状態について情報容量を定義し、体系付けていく。

尚、クエリ 3 は射影によって属性” 都道府県” を除外しているため、テーブル Train に含まれる属性間の関連のうち、“都道府県” に関するものは当然表 5 には含まれていない。しかし、 そもそも表中に表われない属性に関して属性間の関連を議論することは無意味であるので、射影による結果は情報が失われたとは考えない。

3 情報容量

クエリおよびその対象となるデータベースが含む属性集合において、表現可能な属性間の関連を情報容量とする。なお、情報容量は以下に定める情報容量式によって表すものとする。

3.1 情報容量式

定義 3.1 (情報容量式)
情報容量を和積の形で表現した式を情報容量式 $\phi()$ とする
(A,B は任意の積項を表す)

1. 属性間の関連を直接的に表現できる場合、それらの属性の積を取り、積項として式に表す
   但し、 $A * A = A$
   $A * B = B * A$
2. 属性の積項で表される情報容量を複数含む場合、それらの積項の和として式にあらわす
   但し、 $A + A = A$
   $A + B = B + A$

情報容量の表現方法の例を以下に示す。[2]

例 3.1 属性 A, B, C を含む関係 $R(a, b, c)$ の情報容量は
$\phi(R) = a * b * c$

例 3.2 関係 $R(a, b, c)$, $S(d, e, f)$ を結合した情報容量は
$\phi(R \bowtie S) = a * b * c + d * e * f$

例 3.3 表 3 に表現される情報の情報容量は
ϕ(表 3) = t.都道府県 $* t$.駅 $* t$.鉄道会社

表 3 に示される 3 つの属性は、全ての組み合わせにおいてそれらの属性間に関連が表現されているので、情報容量式として 3 つの属性の積を取る。

例 3.4 表 4 に表現される情報の情報容量は
ϕ(表 4) = t.都道府県 $* t$.駅
$+ t$.都道府県 $* t$.鉄道会社

表 4 に示される 3 つの属性は、「“都道府県” と “駅”」「“都道府県” と “鉄道会社”」の間の関連は表現されているが、「“駅” と “鉄道会社”」の間の関連は表現されていない。そこで、情報容量式としては 3 つの属性の積を取り、1 つの積項とするのではなく、関連のある組み合わせで属性の積を取り、それぞれの項を和を全体の式とする。

3.2 結果表示の状態

クエリを実行した表示結果が、そのクエリの対象となるデータベースに含まれる情報を過不足なく的確に反映し、情報が適切に表現されている状態を、**適正表示**と呼ぶ。逆に、クエリ実行後の表示結果において、データベース上に存在する情報が失われている、またはデータベース上に存在しない情報が表示される、といった状態を**表示異常**と呼ぶ。表示異常をさらに細かく分類すると、データベース上には存在する属性間の関連が失われ、表示結果において表現されない状態を**過小表示**、データベース上には存在しない属性間の関連が、表示結果ではあたかも存在するように表現されてしまう状態を**過大表示**と分けられる。また、部分的に過小表示であり、また他の部分では過大表示でもあるという状態を、**過小表示と過大表示の混在状態**と呼ぶ。

クエリおよびその対象となるデータベースの情報容量を比較することにより、結果表示の状態を分類する。

定義 3.2 (結果表示の状態)
データベースの情報容量 $(\phi(DB))$ とクエリの情報容量 $(\phi(Q))$ を比較し、それらの大小関係により、結果表示の状態をそれぞれ以下のように定義する

1. $\phi(DB) = \phi(Q)$ の場合を**適正表示**とする
2. $\phi(DB) \neq \phi(Q)$ の場合を**表示異常**とする
   (a) $\phi(DB) > \phi(Q)$ の場合を**過小表示**とする
   (b) $\phi(DB) < \phi(Q)$ の場合を**過大表示**とする
   (c) $\phi(DB) \not> \phi(Q)$ かつ $\phi(DB) \not< \phi(Q)$ の場合を**過小表示と過大表示の混在**とする

4 情報容量の算出

情報容量を比較することによって表示異常の有無を判定するために、まずクエリおよびデータベースの情報容量を求める必要がある。情報容量は、クエリおよびデータベースに含まれる属性間の関連でありクエリから算出する。

[2] 関係 R,S は非自明な join dependency を含まない

4.1 クエリの情報容量

先に述べたように表示結果における属性間の関連はTFEによって指定される。TFEにはそれぞれの属性に対するレイアウト指定演算子が含まれているが、レイアウトの方向がいかなるものであれ、属性間の関連に影響することはない。よってクエリの情報容量を算出する際には、TFEのネスト構造にのみ着目する。クエリの情報容量を算出する手順を以下に示す。

[クエリの情報容量を求める手順]

1. TFEに含まれる属性およびそのネスト構造を抜き出す
2. 1で求めたものに、以下の規則を適応する

Rule 1 $\phi\ (\ a\ )\ =\ a$

Rule 2 $\phi\ (\ [A]\ )\ =\ \phi\ (\ A\ )$

Rule 3 $\phi\ (\ a\ A\ )\ =\ a\ *\ \phi\ (\ A\ )$

Rule 4 $\phi\ (\ [A]\ [B]\ )\ =\ \phi\ (\ [A]\ )\ +\ \phi\ (\ [B]\ )$

Rule 5 $\phi\,(a\ [A]\ [B]) = a{*}\ \phi\,([A]) + a{*}\ \phi\,([B])$

(a は任意の属性、A,B は任意の TFE を表す)

4.2 データベースの情報容量

クエリの対象となるデータベースとは、FROM句に指定されるテーブルを示す。また指定されたテーブルが複数である場合、それらの結合条件がWHERE句に指定される。先に述べたように射影は情報容量の変化に関与しないものと考えるので、対象データベースに含まれる属性はテーブルに含まれる属性ではなく、クエリのTFEに含まれる属性とする。したがって、データベースの情報容量を求めるには、クエリのTFEに加え、WHERE句に含まれる属性にも着目する。ここで着目したいのは属性間の関連であるので、WHERE句を構成する条件文の中でも、ある属性に対して定数や値の範囲を指定するものは除外し、結合条件を示すもののみを考慮する。以下にデータベースの情報容量を求める手順をまとめる。

[データベースの情報容量を求める手順]

1. TFEに含まれる属性に着目
   同一テーブルの属性を全て1つの積項にまとめ、それぞれのテーブルの積項の和を取る
2. WHERE句に含まれる属性に着目[3]
   条件文の左辺または右辺の属性に
   (a) テーブルの主キーがある場合
   条件文で結ばれる2つのテーブルの項(1で求めたもの)の積を取り、1つの積項にまとめる
   (b) TFEに含まれる属性がある場合
   その属性と、条件文の他辺の属性のテーブルの項(1で求めたもの)の積を取る

4.3 情報容量算出の例

ここで、先に述べた手順に基づいて、クエリ文からクエリおよびその対象となるデータベースの情報容量を算出する例を示す。

例 4.1 前出のクエリ *3* より、クエリおよびデータベースの情報容量を求める

< クエリの情報容量 >

1. TFEから属性およびそのネスト構造を抜きだし
   [t.駅 [t.鉄道会社 h.名称]]　を得る
2. 1で求めたものに、定められた5つの規則を適応させると、以下のようになる。

$$\begin{aligned}
\phi(Q) &= \phi(\ [\ t.\text{駅}\ [t.\text{鉄道会社}\ \ h.\text{名称}]\ ]) && (1)\\
&= \phi(\ t.\text{駅}\ [t.\text{鉄道会社}\ \ h.\text{名称}]\ ) && (2)\\
&= t.\text{駅} * \phi(\ [t.\text{鉄道会社}\ \ h.\text{名称}]\ ) && (3)\\
&= t.\text{駅} * \phi(t.\text{鉄道会社}\ \ h.\text{名称}) && (4)\\
&= t.\text{駅} * t.\text{鉄道会社} * \phi(h.\text{名称}) && (5)\\
&= t.\text{駅} * t.\text{鉄道会社} * h.\text{名称} && (6)
\end{aligned}$$

まず第一段階において、TFEより情報容量算出に必要の無い部分を除外する。この例の場合、結合子および反復子の反復方向指定を除外するだけでよい。

第二段階を式変換に沿って解説する。(1)は$\phi([A])$に置き換えることができるので、Rule2を適用する

[3] 他の場合分けに関しては今後の課題である

と、(2) に変換でき、(2) は $\phi(a\ A)$ に置き換えられるので Rule3 を適用し、(3) に変換できる。(3) の $\phi(\ )$ の部分は $\phi([A])$ に置き換えられるので、Rule2 を適用すると、(4) に変換できる。(4) の $\phi(\ )$ の部分は $\phi(a\ A)$ に置き換えられるので、Rule3 を適用すると、(5) に変換できる。(5) の $\phi(\ )$ の部分は、$\phi(a)$ に置き換えられるの Rule1 を適用すると、 最終的な情報容量式として (6) が求められた。

＜データベースの情報容量＞

1. TFE に含まれる属性に着目し、同一テーブルの属性を全て 1 つの積項にまとめ、
   t.駅 $*$ t.鉄道会社　と　h.名称　を得る
   それらの和を取り　t.駅 $*$ t.鉄道会社 $+$ h.名称 を得る

2. WHERE 句において、左辺の属性 “t. 駅” が TFE に含まれるので、1 で求めた式の項 “h. 名称” と、“t. 駅” の積を取り、項 “h.名称 $*$ t. 駅” を得る。

最終的に、データベースの情報容量として

$\phi(DB)\ =\ t$.駅 $*$ t.鉄道会社 $+$ h.名称 $*$ t.駅

を算出できた。

まず第 1 段階として TFE に含まれる属性を同一テーブルごとに分類する。この例では指定されたテーブルが Station および Hotel の 2 つであるので、2 つの積項を得る。これらの和を取り、1 つの式にまとめる。次に第 2 段階として、WHERE 句に着目する。その条件文の両辺の属性共にそれぞれのテーブルのキーではないが、左辺の属性 “t. 駅” が TFE に含まれるので、手順 2(b) に従い、1 で求めた式の中で、条件文の右辺 “h. 所在地” のテーブル Hotel の項にあたる “h. 名称” と、“t. 駅” の積を取る。以上により、最終的な情報容量式が求められた。

4.4　情報容量の比較

情報容量の比較は、クエリ、データベースそれぞれの情報容量式に含まれる項の単位で行う。以下に項単位の大小判定基準および表示結果の状態判定基準をまとめる。

[**項単位の大小判定基準**]
(A,B は任意の積項を表す)

1. 等価関係 $(A = A)$
2. 大小関係 $(A < A * B)$
3. 比較不可能
   (1,2 以外の関係 ($\neq$ かつ $\not<$ かつ $\not>$)

[**項集合の判定基準**]
(α, β を情報容量の項の集合とする)
$^{\forall i ^{\exists} j}\ (\alpha_i \leq \beta_j)$　のとき　$\alpha \subset \beta$

[**表示結果の状態判定基準**]
クエリ、データベース それぞれの情報容量式を以下のように表し、それぞれの式の項を要素とする集合を **Q**,**DB** とすると、

$\phi(Q) = Q_1 + Q_2 + \cdots + Q_m$
$\phi(DB) = DB_1 + DB_2 + \cdots + DB_n$

1. $\mathbf{Q} \subset \mathbf{DB}$ かつ $\mathbf{Q} \supset \mathbf{DB}$　のとき適正表示
2. $\mathbf{Q} \subset \mathbf{DB}$ かつ $\mathbf{Q} \neq \mathbf{DB}$　のとき過小表示
3. $\mathbf{Q} \supset \mathbf{DB}$ かつ $\mathbf{Q} \neq \mathbf{DB}$　のとき過大表示
4. $\neg(\mathbf{Q} \subset \mathbf{DB})$ かつ $\neg(\mathbf{Q} \supset \mathbf{DB})$ のとき過小表示と過大表示の混在

例 4.2 前出のクエリ 3 より求められたクエリおよびデータベースの情報容量を比較し、表示結果の状態を判定する。

例 4.1 より、クエリおよびデータベースの情報容量は以下のように求められた。

$\phi(Q)\ =\ t$.駅 $*$ t.鉄道会社 $*$ h.名称
$\phi(DB)\ =\ t$.駅 $*$ t.鉄道会社 $+$ h.名称 $*$ t.駅

このとき、全ての DB_i に対し

$DB_1 = t$.駅 $*$ t.鉄道会社 $<$
(t.駅 $*$ t.鉄道会社) $*$ h.名称 $= Q_1$

$DB_2 = h$.名称 $*$ t.駅 $<$
(t.駅 $*$ t.鉄道会社) $*$ h.名称 $= Q_1$

となる Q_j が存在するので、$\mathbf{DB} \subset \mathbf{Q}$ となる。かつ、$\mathbf{Q} \neq \mathbf{DB}$ であるので、表示結果の状態判定基準 (3) にあてはまり、クエリ 3 は過大表示であると判定できる。

5　システムの概要

5.1　システム構成

以上に述べた理論に基づき、SuperSQL クエリ作成支援システムのプロトタイプを実装した。システムの構成図を以下の図 1 に示す。

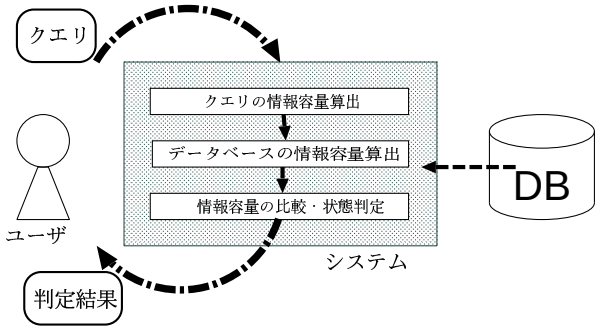


図 1: システム構成図

本システムの目的は、クエリ作成時にクエリの情報容量およびデータベースの情報容量を比較することによって表示異常の有無を判定することである。つまり、ユーザはクエリに対して本システムを実行することで、そのクエリによる表示異常の有無を判定することが可能となっている。

本システムは大きく分けて三段階で構成されている。まず第一段階として、本システムは入力として得たクエリから、クエリの情報容量を算出する。次に第二段階では入力として得たクエリに加え、クエリの対象となるデータベースから取りだしたスキーマ情報などを考慮し、データベースの情報容量を算出する。最後に第三段階で、前段階で算出されたクエリおよびデータベースの情報容量を比較し、入力されたクエリによる表示結果の状態判定を行う。そして、その判定結果をシステムの出力としてユーザに返す。

5.2 システム出力例

ではここで、システムの出力例を示す。

例 5.1 前出のクエリ *3* に対し、システムを実行した結果

< システム出力 >

$\phi(Q)$ = t.駅 $*$ t.鉄道会社 $*$ h.名称
$\phi(DB)$ = t.駅 $*$ t.鉄道会社 + h.名称 $*$ t.駅
表示結果は**過大表示**です。
$\phi(Q)$ の t.駅 $*$ t.鉄道会社 $*$ h.名称 に対し、
$\phi(DB)$ には t.駅 $*$ t.鉄道会社 と h.名称 $*$ t.駅 の関連しか存在しない

システム出力より、ユーザは、“t. 鉄道会社” と “h. 名称” の間の関連が増えてしまっていることが分かる。そこでクエリ 3 を見ると、それらの属性が TFE において並列に連結されていることが読み取れる。つまり、その部分のネスト構造を変更することによって、クエリを修正できるということが分かる。

6 まとめ

SuperSQL クエリおよびその対象となるデータベースが含む属性集合において、表現される属性間の関連を情報容量として定義し、クエリおよびデータベースの情報容量の大小関係による表示結果の状態の分類を行った。また、クエリ文からクエリとデータベースの情報容量を算出する手順を確立し、それらの情報容量式を比較して大小判定を行う基準を定めた。これによりクエリ作成時に表示結果における表示異常の有無を判定し、結果をユーザに提示するシステムを実現した。

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: http://www.db.ics.keio.ac.jp/ssql

[3] 遠山 元道, “ターゲットリストの拡張によるデータベース出版と概視の実現”: 信学技報, 電子情報通信学会,Vol.93,No.152,pp. 79-88,1993

[4] 遠山 元道, “データベース出版における木構造スキーマの情報容量”: 情処研報, 情報処理学会,Vol.98,No.58, pp. 173-180,1998

[5] F.P.Preparata/R.T.Yeh 著, 榎本 彦衛 訳, “離散構造入門”: 日本コンピュータ協会, pp. 210-250,1980

[6] 笹田 麻衣子, 遠山 元道, “SuperSQL の HTML 出力における時間連結子の実装”: 日本データベース学会 *Letters*,Vol.2, No.1, pp. 135-138, 2003