SuperSQL クエリ作成支援系における情報容量の提示

Information Capacity Presentation in the SuperSQL Query Support System

大澤 幸子† 遠山 元道‡
Sachiko OSAWA Motomichi TOYAMA

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する 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．

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 the 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 technique 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.

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[3]．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好ましくない．

今後SuperSQL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[4] 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 レイアウト演算子

SperSQL はクエリにおいて，表 1 に示す連結子，反復子によって表示結果のレイアウト指定を行うことが可能である．

表 1 レイアウト演算子
Table 1 Layout operatiors

,	水平連結子
!	垂直連結子
%	深度連結子
[],	水平反復子
[]！	垂直反復子
[]%	深度反復子

2.2 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関係を表 2,3 に示す．

表 2 関係 Train
Table 2 Relation Train

都道府県	駅	鉄道会社
東京都	渋谷	ＪＲ
東京都	渋谷	東京急行
東京都	品川	ＪＲ
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 3 関係 Hotel
Table 3 Relation Hotel

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

表 2 の関係 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 の関係 Hotel は，「どの“所在地”になんという“名称”のホテル

† 学生会員 慶應義塾大学大学院理工学研究科修士課程
sachi@db.ics.keio.ac.jp
‡ 正会員 慶應義塾大学理工学部情報工学科
toyama@ics.keio.ac.jp

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，全て関係 Train の属性“駅”の値となっている．

表2,3に対し,次のクエリ１を実行した結果を表４に示す.

＜クエリ 1＞ GENERATE latex
[t.駅,[t.鉄道会社, h.名称]!]!
FROM Train t, Hotel h
Where t.駅 = h.所在地

表４ クエリ１の結果
Table 4 Result of the query1

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

クエリ1は関係Trainの“駅”と関係Hotelの“所在地”を結合条件としているため，表4ではそれぞれの関係に含まれる「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1の対象となるデータベース，つまり上記2つの関係には含まれない「“鉄道会社”と“名称”」の間の関連が存在しているようにみえる．これら3つの属性を1つの表に表す場合，クエリ2に示すように“駅”に対して“鉄道会社”および“名称”を個別にグルーピングすることで，表5のように適正な表示結果を得ることができる．

＜クエリ 2＞ GENERATE latex
[t.駅,[t.鉄道会社]!, [h.名称]!]!
FROM Train t, Hotel h
Where t.駅 = h.所在地

表５ クエリ２の結果
Table 5 Result of the query2

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

このようにクエリの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，また逆に，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，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，このクエリは射影によって属性“都道府県”を除外しているため，関係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4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．情報容量に関する代表的な研究として，R.Hull らによるものが挙げられる[5].

3.1 情報容量式

[定義 3.1]（情報容量式）
属性集合αの情報容量を属性の和積の形で表現した式を情報容量式φ(α)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表す
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[1].

[例 3.1] 属性 A, B, C を含む関係 R(a, b, c)の情報容量は
φ(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
φ(R ⋈ S) = a＊b＊c＋d＊e＊f
[例 3.3] 表 4 に表現される情報の情報容量は
φ(表 4) = t.駅＊t.鉄道会社＊h.名称
表 4 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存在するので，情報容量式として 3 つの属性の積を取る．
[例 3.4] 表 5 に表現される情報の情報容量は
φ(表 5) = t.駅＊t.鉄道会社 ＋t.駅＊h.名称
表 5 に示される 3 つの属性のうち，互いに関連を持つのは「“駅”と“鉄道会社”」および「“駅”と“名称”」の 2 組であり，「“鉄道会社”と“名称”」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表現される属性間の関連が，データベース上に存在する関連と一致しない，といった状態を**表示異常**と呼ぶ．表示異常をさらに分類し，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いる状態を**過大表示**とする．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，

[1] 関係 R,S は第五正規形とする

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φ(DB)とクエリの情報容量φ(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） φ(DB)＝φ(Q) の場合を**適正表示**とする

（2） φ(DB)≠φ(Q) の場合を**表示異常**とする

（3） φ(DB)＞φ(Q) の場合を**過小表示**とする

（4） φ(DB)＜φ(Q) の場合を**過大表示**とする

（5） φ(DB)≮φ(Q)　かつ　φ(DB)≯φ(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.

4.1 クエリの情報容量

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求める

（2）(1)の結果に，以下の規則を適用する

[Rule 1] φ(a) = a

[Rule 2] φ([A]) = φ(A)

[Rule 3] φ(a A) = a＊φ(A)

[Rule 4] φ([A][B]) = φ([A]) + φ([B])

[Rule 5] φ(a[A][B]) = a＊φ([A]) + a＊φ([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による情報量の変化は議論の対象としないものとするので，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全属性ではなく，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[2]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求め

[t.駅 [t.鉄道会社 h.名称]] を得る

（2）(1)の結果に，定められた 5 つの規則を繰り返し適用すると，以下のようになる.

φ(Q) = φ([t.駅 [t.鉄道会社 h.名称]]) (1)

= φ(t.駅 [t.鉄道会社 h.名称]) (2)

= t.駅＊φ([t.鉄道会社 h.名称]) (3)

= t.駅＊φ(t.鉄道会社 h.名称) (4)

= t.駅＊t.鉄道会社＊φ(h.名称) (5)

= t.駅＊t.鉄道会社＊h.名称 (6)

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φ([A])に置き換えることができるので，Rule2 を適用すると，(2)に変換でき，(2)はφ(aA)に置き換えられるので Rule3 を適用し，(3)に変換する．(3)のφ( )の部分は Rule2 を適用すると，(4)に変換できる．(4)のφ()の部分は Rule3 を適用すると，(5)になり，(5)のφ( )の部分は，Rule1 を適用すると，最終的な情報容量式として(6)が求められる.

＜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以下を得る.

φ(DB) = t.駅＊t.鉄道会社 + h.名称＊t.駅

第一段階として 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. 次に第二段階として，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”の関係 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る.

[2] データベースは第五正規形であるとする

[3] 他の場合分けに関しては今後の課題である

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) 比較不能 (1,2 以外の関係(≠ かつ ≯ かつ ≮))

[項集合の判定基準] (α,βを情報容量の項の集合とする)

$\forall i \exists j (\alpha_i \leqq \beta_j)$ 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，それぞれの式の積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) **Q**⊂**DB** かつ **Q**⊃**DB** のとき適正表示
(2) **Q**⊂**DB** かつ **Q**≠**DB** のとき過小表示
(3) **Q**⊃**DB** かつ **Q**≠**DB** のとき過大表示
(4) ¬(**Q**⊂**DB**) かつ ¬(**Q**⊃**DB**) のとき過小表示と過大表示の混在}

[例 4.2] 前出のクエリより求められたクエリ 1 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 1 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，**DB**⊂Q となる．かつ，**Q**≠**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 1 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

以上に述べた手続きに基づき, SuperSQL クエリ作成支援システムのプロトタイプを実装した．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．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 1 に対し，システムを実行した結果

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．そこで前出のクエリ 1 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．よって，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．つまり，クエリ 1 をクエリ 2 に修正することで，表 5 に示すような適正な表示結果を得ることができる．

＜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

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．

現状では，クエリの対象となるデータベースは第五正規形であることが制約条件となっている．また，データベースの情報容量算出手順については，対応できるクエリが限られている．今後これらの制約を緩和していくことが課題である．

[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, “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
[4] 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
[5] R. Hull, “Managing Semantic Heterogeniety in Database”, A Theoretical Perspective, in Proc.PODS ’97, ACM, pp51-61, 1997

大澤 幸子 Sachiko OSAWA
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．