データの連続性にもとづく検索言語ＳＱＬ-ＳＦの提案

Proposal of ”SQL-SF”, a Query Language Based on Data Continuity

嶋村　勇志†　　遠山　元道††

†††慶應義塾大学理工学部情報工学科　〒 223–8522 神奈川県横浜市港北区日吉 3–14–1
E-mail: †shima@db.ics.keio.ac.jp, ††toyama@ics.keio.ac.jp

あらまし　リレーショナルモデルでは、データの集合をリレーションとしてモデル化しているため、個々のデータの連続性、例えば、ある一連のデータが条件に連続して合致しているかどうかや、前後のタプルとの大小関係にもとづいた検索を行うことができない。そこで、本論文ではリレーショナルモデルに順序の概念を取り込んだデータモデルである SR モデルとともに、そのモデルに従って定義した検索言語である SQL-SF(SQL with Sequential Function) を提案する。

キーワード　DB 言語、問合せ処理、情報検索

Takeshi SHIMAMURA† and Motomichi TOYAMA††

†††Department of Information and Computer Science, Faculty of Science and Technology,
Keio University
Hiyoshi3–14–1, Kouhoku-ku, Yokohama-shi, Kanagawa, 223–8522 Japan
E-mail: †shima@db.ics.keio.ac.jp, ††toyama@ics.keio.ac.jp

Abstract　The relational model is a data model which represents set of data as relation. Therefore, it is not aware of retrieval based on continuity of data, for example, if a series of data successively satisfies a condition. For another example, it cannot write a condition comparing with tuples before or after the tuple while there is no idea of ’before’ nor ’after’ in the relational model. In this paper, we propose *SR Model (Sequence and Relation Model)*, in which we add an idea of sequence into the relational model, and the query language based on the model, *SQL-SF (SQL with Sequential Function)*.

Key words　DB Language, Query Management, Information Retrieval

1. はじめに

従来のリレーショナルモデルでは、データの集合をリレーションとして表現し、その連続性、データの順序は意味を持たなかった。実際に SQL によって検索を行う際には、ORDER BY 句を使用することにより、アプリケーションに出力する時点においては明示的にタプルの順序を指定することができるが、その順序を検索の条件として利用することはできなかった。

そこで本研究では、タプルの連続性にもとづいたクエリを記述可能にする SQL 拡張言語である、SQL-SF(SQL with Sequential Function) を提案すると共に、その言語モデルである SR モデルを提案する。

本稿の構成は以下の通りである。まず、2. 章で関連研究について述べ、3. 章では SQL-SF の言語仕様と、従来の SQL との違いを示す。4. 章ではその SQL-SF を用いたクエリ例を挙げ、その表現力を示し、5. 章では SQL-SF の基盤となるモデルである SR モデルの概要を説明する。また、6. 章では実装したプロトタイプシステムの概要を述べ、最後に 7. 章でまとめを述べる。

2. 関連研究

2.1 SQL の拡張言語

データの連続性を用いた検索を可能にするために SQL を拡張した検索言語は、過去に幾つか提案されてきた。

2.1.1 SRQL

Ramakrishnan らの SRQL [7] は、Seshadri らの SEQ システム [9] における検索言語である SEQUIN を改良したもので、データを、リレーション (relation)、グループ化属性 (grouping attributes)、順序付け属性 (sequencing attributes) の３つで表現する。また、リレーショナル代数には、新たに Shift 演算子および Shift All 演算子を追加定義している。本研究では、SRQL と同様なアプローチでリレーショナルモデルの純粋な拡張モデルとして、SR モデルを提案している。

2.1.2 SQL-TS

SQL-TS(Simple Query Language with Time Series) [8] は株価

のデータなどから変動パターンを検索するといった、パターン検索に特化した SQL 拡張言語である。正規表現のように、*(アスタリスク) を用いて繰り返しを表現することにより、より複雑なパターンを条件として検索することができる。また、パターン検索の最適化アルゴリズムとして、KMP アルゴリズム [2] を改良した OPS アルゴリズムを提案している。

しかしながら、SQL-TS では、データの連続性に対応したデータモデルは存在しない。

2.1.3 AQuery

Lerner らの AQuery [4] は KX systems における検索言語である、KSQL [3] の派生言語である。KSQL から、*Arrable* の概念を受け継ぎ、文法をより SQL に近いものにしている。*Arrable* とは、属性の値を、値の集合ではなく配列で表現したもので、AQuery では、データを全てこの *Arrable* で表現する。

AQuery では、検索を行うためのリレーショナル代数を集合ではなく、*Arrable* の代数として再定義している。この点で、データの連続性に関する演算を、リレーショナルモデルに対し拡張定義した SR モデルとはアプローチが異なる。

2.2 SQL の拡張以外のアプローチ

SQL とは全く違う言語仕様で連続的なデータに対する検索を行うアプローチも存在する。AQL [5] や AML [1] はその例である。これらは CDF フォーマットのデータやビットマップイメージに対する検索において有用性を示している。しかし、これらの手法はデータが格納されているそのままの順番に対しクエリを実行するというもので、一般的なデータ検索に応用することは難しいといえる。

3. SQL-SF

本研究で提案する SQL-SF は、SQL の純粋な拡張言語である。従来の SQL の言語仕様に対し、SEQUENCE ON 句、ABOUT 句、SUCH 句を加え、また、グルーピングキーワードである CONSECUTIVES 及び、特殊タプル変数を追加定義した。

本項では、SQL-SF (SQL with Sequential Function) の言語仕様を説明するために、以下のようなサッカー日本代表の試合結果に関するデータベースの例を用いる。

```
試合結果（日付 date,
          対戦相手国 varchar,
          得点 int,
          失点 int）
```

以下に SQL-SF のクエリ例を示す。(注1)

クエリ 3.1
日本代表の試合において、ある対戦相手国に対して連勝した一連の試合結果

```
SELECT * FROM 試合結果
SEQUENCE ON 日付
ABOUT 対戦相手国
SUCH 得点 > 失点
GROUP BY CONSECUTIVES;
```

3.1 SEQUENCE ON

SQL-SF では、まずタプルを SEQUENCE ON 句で指定された属性についてソートする。これは SQL-TS [8] や SRQL [7] の SEQUENCE BY 句における演算と同様であるが、両者と異なる点は、演算が WHERE 句による選択演算の後に実行される点である。クエリ 3.1 の例では試合結果のタプルが試合が行われた日付ごとに順番付けされる。

属性は複数指定することが可能で、その場合は SEQUENCE ON 句で先に指定された属性から優先してソートされるのは、SQL における ORDER BY と同様である [6]。ここで指定されたタプルの順番は、あくまで検索に利用するためであって、必ずしも結果の表示のときの順番とは限らない。出力結果の表示の順序指定は従来通り ORDER BY 句によって指定する。

3.2 ABOUT

ABOUT 句は、SQL-TS の CLUSTER BY 句 [8] や SQL99 の PARTITION BY 句 [6] と同様に、SEQUENCE ON 句の対象となるタプルごとにグループ分けを行う。つまり、ABOUT 句で指定された属性（クエリ 3.1 では「対戦相手国」）ごとにグループ化を行い、その各グループごとに SEQUENCE ON 句で指定された属性によるソートが行われる。もし ABOUT 句が指定されない場合には、タプル全体を１つのグループとして演算を行う。

3.3 SUCH

SUCH 句は SQL における WHERE 句と同様に、関係代数における選択演算を行う。WHERE 句との違いは、条件式の評価をソート後に行うため、SEQUENCE ON 句によって指定された順番にもとづいた選択を行うことができる、という点である。タプルの前後関係に基づいた比較演算を行うために、SQL-SF では４つの特殊タプル変数を定義した。

（1） FIRST
ABOUT 句で指定されたグループ中の最初のタプル
（2） LAST
ABOUT 句で指定されたグループ中の最後のタプル
（3） PREV
ABOUT 句で指定されたグループ中で、現在評価の対象となっているタプルの一つ前のタプル
（4） NEXT
ABOUT 句で指定されたグループ中で、現在評価の対象となっているタプルの一つ後のタプル

具体的な使用法については 4.2 節で述べる。ここで、特殊タプル変数によって指定したタプルが存在しない場合（例えばグループ中の一番最初のタプルを評価しているときに PREV が指定された場合）には、全ての属性に対し、null が返される。

これらの特殊タプル変数は、SUCH 句および、SELECT 句に

(注1)：従来の SQL では、一般的に GROUP BY が指定されているときに SELECT 句に*は指定できないが、SQL-SF では可能である。これについては、6.3 節において、後述する。

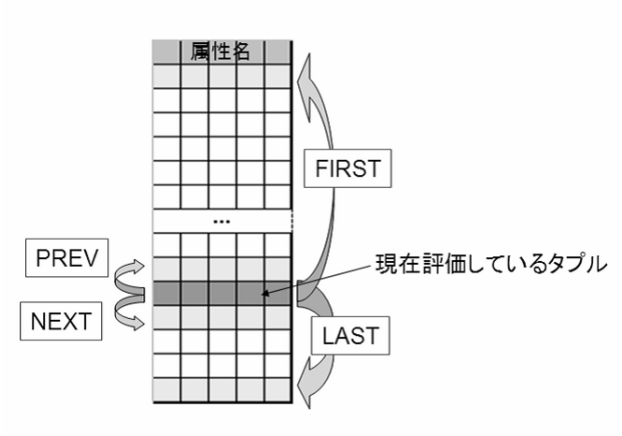

図 1　特殊タプル変数

おいて指定可能である。

3.4　CONSECUTIVES

SQL-SF では、GROUP BY 句において、従来通りの属性のカンマリストの他に、グルーピングキーワードである CONSECUTIVES を指定することが可能である。CONSECUTIVES を指定すると、SUCH 句の条件に連続して合致する一連のタプルごとにグループ化することができる。クエリ 3.1 の例では、図 2 のように対戦相手国ごとに分けた各グループにおいて、SUCH 句の条件が「得点 > 失点」、つまり連勝中である一連のタプルごとにグループ分けする。

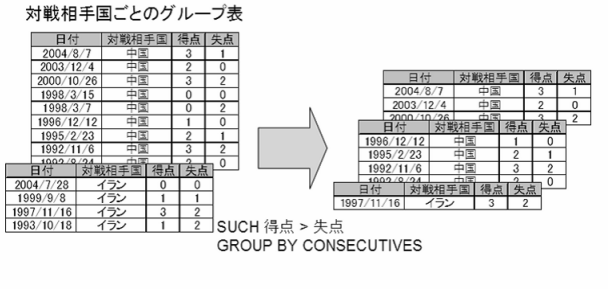

図 2　GROUP BY CONSECUTIVES の実行例

CONSECUTIVES の指定により、「ある条件が連続して起こる」といった検索が可能になる。

3.5　言 語 仕 様

以上の仕様をまとめて BNF 記法によって表現したものを以下に示す。

SELECT <*| 属性名リスト >
FROM < 表名 >[{, < 表名 >}…]
[WHERE < 選択条件 > [{,< 選択条件 >}]]
[SEQUENCE ON < 属性名リスト >]
[ABOUT < 属性名リスト >]
[SUCH < 選択条件 >]
[GROUP BY < 属性名リスト > |CONSECUTIVES]
[HAVING < 選択条件 >]
[ORDER BY < 属性名リスト >]

4.　ク エ リ 例

4.1　サッカー日本代表データベース

3. 章と同様のサンプルスキーマを用いたクエリの例を以下に示す。

クエリ 4.1
日本代表の試合で 3 連勝以上した一連の試合を検索

```
SELECT * FROM 試合結果
SEQUENCE ON 日付
SUCH 得点 > 失点
GROUP BY CONSECUTIVES
HAVING count(*) >= 3
ORDER BY count(*);
```

クエリ 4.1 は、クエリ 3.1 に続いて GROUP BY CONSECUTIVES を使用する例である。HAVING 句と組み合わせることにより、「得点 > 失点」という SUCH 句の条件が、n 以上繰り返されている、即ち「n 連勝以上」といった条件指定が可能になっている。ここで、比較演算子を変更すれば、「n 回以下 (count(*) <= n)」、または「n 回 (count(*) = n)」繰り返されている、などといった指定も可能である。

4.2　アクセス解析

一般的な Web ページのアクセス解析では、日ごと、時間帯ごとのアクセス数集計やリファラーの集計などが主に行われている。これは従来の SQL の集計関数 (Aggregate Function) を用いれば、容易にクエリを記述することができる。これに加えて SQL-SF ではユーザの閲覧パターンを検索することができる。ここではある Web サイトにおけるアクセスログから抽出した以下の様なサンプルスキーマを想定した。

```
access（id int,
        requestdate date,
        requesttime time,
        hostname varchar,
        filename varchar ）
```

ここで、簡単のためホスト名 (hostname 属性) とユーザは 1 対 1 に対応するものとして話を進める。

クエリ 4.2
toppage.html->diary.html という閲覧パターンを日ごとに集計

```
SELECT requestdate, count(*) FROM access
WHERE filename LIKE '%html'
SEQUENCE ON requestdate, requesttime
ABOUT hostname
SUCH PREV.filename = 'toppage.html'
        AND filename = 'diary.html'
GROUP BY requestdate;
```

Web サイトではそのメニューやリンクの配置が、ユーザの閲覧パターンを大きく左右する。クエリ 4.2 のような検索を行うことで、toppage.html のデザインを変更した前後で、それがユーザの閲覧パターンにどのように影響したかを読み取ることができる。

クエリ 4.2 では特殊タプル変数が使用されている。SUCH 句において、

特殊タプル変数. 属性名

のように、前後のタプルの属性を参照することが可能である。尚、もし属性名を指定する際に、複数のテーブルに同名の属性が存在していて、テーブル名を明示的に指定する必要があるときは、

特殊タプル変数. テーブル名. 属性名

という記述になる。3.3 節でも触れた通り、特殊タプル変数が存在しないタプルを示す場合には、全属性に対して null が返される。

SEQUENCE ON 句では requestdate、requesttime の二つの属性が指定されているが、requestdate が優先的にソートされるのは、3.1 節で述べた通りである。

クエリ 4.3

サイト閲覧者が前回サイトを閲覧してからの経過時間を検索

```
SELECT hostname, PREV.time − time
FROM access
WHERE filename LIKE '%html '
SEQUENCE ON date, time
ABOUT hostname
SUCH PREV.time − time > 15min
```

ユーザの閲覧頻度を知ることで、サイト管理者は最適なサイト更新の頻度を決定する参考にすることができる。クエリ 4.3 では、前回の訪問から次の訪問までに要した時間を検索することで閲覧頻度を解析している。ここで、15 分以上の間隔を独立した閲覧とみなしている。

クエリ 4.3 の例では、SELECT 句においても、特殊タプル変数を指定できることを示している。

5. SR モデル

リレーショナルモデルでは、リレーションを表として表現し、行をタプル、列を属性と表現する 2 次元の構造を持っている。表は行の論理的な集合として表現されるが、ここで行には順番がなく、また列にも順番はない。そこで SR モデル (Sequence and Relation Model) では、従来の 2 次元のテーブルに、順番のある「シーケンス軸」を新たに定義した。(図 3)

5.1 S 表への変換

SR モデルでは、従来のリレーショナルモデルにおける表 (Table) は属性軸とタプル軸による 2 次元の平面で表される。ここでこれらの表を R 表と呼ぶことにする。この R 表に対し、

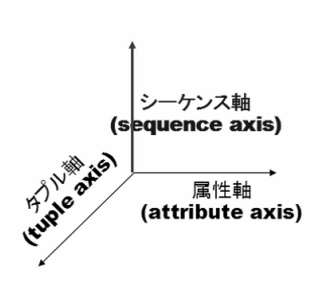

図 3　SR モデル

順番が定義されると、順番のないタプル軸に並べられたタプルが、順番のあるシーケンス軸に昇順に射影される。これによって生成された属性軸、シーケンス軸による 2 次元の平面を、R 表に対して S 表と定義する。SR モデルでは、データを R 表と S 表の両方を用いて表現する。

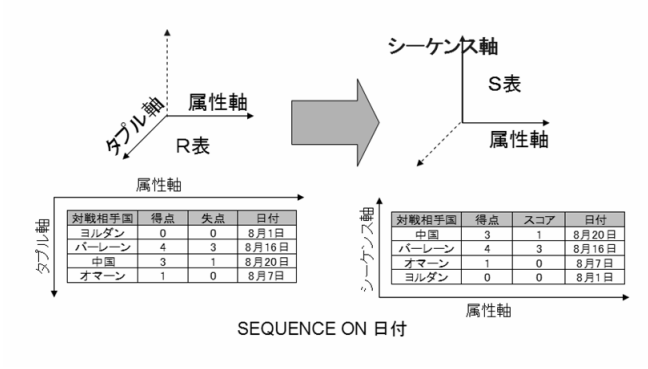

図 4　S 表への変換

SQL-SF では、タプルの順序を SEQUENCE ON 句で指定された属性によって定義する。クエリ 3.1 の例では、SEQUENCE ON 句に「日付」属性が指定されているので、タプルが日付の早いタプルから順にシーケンス軸へ射影されることになる。(図 4)

5.2 グループ化

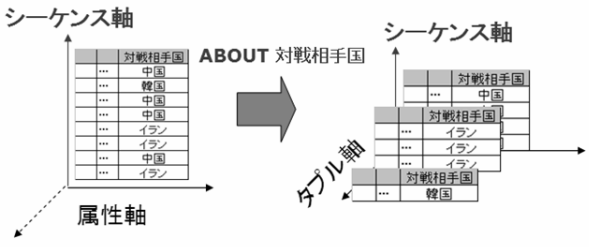

図 5　S 表のグループ化

リレーショナルモデルにおけるグループ化とは、タプルをある属性の組の値が等しい行を同じグループとしてグループ分けすることである。SQL で GROUP BY 句が指定されると、2 次元の表は、GROUP BY 句で指定された属性ごとに集計され、グループ表を生成する。また、グループを選択するには、HAVING 句において属性名や集計関数を用いた条件指定をす

ることが可能であった。しかし、HAVING 句における条件指定はグループに対してのみ適用可能であり、グループ化後に個々のタプルに対する選択演算を行うことができなかった。データの連続性を利用した検索では、タプルをグループごとに分けた後、各グループにおいて、個々のタプルに対して選択を行うことができる必要がある。

そこで SR モデルにおいて、グループ化とは図 5 のように、タプルをある決まりに従ってグループ分けし、それをタプル軸に並べるものとする。これによって、グループ化した後も、属性-シーケンス平面において各タプルの選択演算が可能であり (SUCH 句で与えられた条件)、かつ属性-タプル平面において、従来通りグループごとの選択演算も行うことができる。(HAVING 句で与えられた条件) ここで、各グループは、順番の関係ない軸であるタプル軸に並べられているため、グループ間に前後関係は存在しない。

例えば、クエリ 3.1 は日本代表の試合において、ある対戦相手国に対して連勝した一連の試合結果を検索したい、というクエリであったが、日付による S 表を ABOUT 句で指定されている「対戦相手国」属性ごとにグループ化した後、各グループごとに「得点 > 失点」の条件による選択を実行する。

5.3 R 表と S 表の結合

5.3.1 R 表と S 表の直積演算における順序の保存

5.1 節で述べたように、R 表に順番が定義されると、S 表に変換される。ここで、S表はデータの順序に大きな意味があるため、S 表と R 表の直積演算の結果その順序が失われてしまっては都合が悪い。そこで、SR モデルでは直積演算を以下のように定義する。

定義 5.1

R 表と *S* 表の直積演算の結果

（1） $R_A \times R_B = \{R(r_A, r_B) | r_A \in R_A \wedge r_B \in R_B\}$

（2） $S_A \times R_B = \{\sum_{i=1}^{|S|} S(S_A i, r_B) | r_B \in R_B\}$

（3） $R_A \times S_B = \{\sum_{i=1}^{|S|} S(r_A, S_B i) | r_A \in R_A\}$

（4） $S_A \times S_B = \{\sum_{i=1}^{|S|} \sum_{j=1}^{|S|} S(S_A i, S_B j)\}$

ここで、S_i とは、S 表の要素における i 番目の要素を指す。

定義 5.1 に示すように、R 表同士の直積演算は、従来のリレーショナルモデルの通り、R_A の要素と、R_B の要素を組み合わせて得られる全てのタプルである。また、R 表と S 表の直積でも同様に両表の要素を組み合わせて得られる全てのタプルであるが、ここで、S 表におけるタプルの順序は保存される。さらに、S 表同士の結合では、S_A の順序が優先的に保存される。

5.3.2 結合演算の定義

R表同士（仮に R_A、と R_B とする）の θ-結合とは R_A と R_B の直積集合のタプルの中で、R_A からの属性 A_i と R_B からの属性 B_j が θ の関係にあるタプルの集合だけからなるリレーションである。つまり、結合演算とは、直積演算 (Cartesian Product) と選択演算 (Selection) を用いて表現できる演算である。

5.3.1 節でも述べたように、S 表では、データの前後関係に意味があるため、タプルが一つでも失われてしまうと都合の悪い場合が多い。しかし、表同士の結合演算は、上述した通り、直積演算と選択演算の組み合わせで表現されるため、S 表に含まれるタプルに対応するタプルが結合する R 表に存在しなければ、S 表のタプルの情報、つまりその連続性が失われてしまう可能性がある。

そこで、SR モデルでは、結合演算を以下のように定義する。

（1） R表同士の結合は従来の関係モデルにおける結合演算と同様である。

（2） S 表と R 表の結合は以下のように定義する。

定義 5.2

$S \bowtie_{A_i \theta B_j} R$

$S \bowtie_{A_i \theta B_j} R = T \cup (S' \times D)$

$T = A_i \theta B_j$

$S' = S - T[A_1, A_2, \cdots, A_n]$

$D(A_1, A_2, \cdots, A_n)$:全ての属性値が *null* 値の 1 タプルからなる定値リレーション

ただし、*S* $[A_1, A_2, \cdots, A_n]$, *R* $[B_1, B_2, \cdots, B_m]$

これによって、S に対応する R の要素が存在しなかった場合には、S と全ての属性値が null 値の定値リレーション D との直積演算が行われる。ここで、S 表のタプルの順序は保存されるものとする。

（3） R表とS表の結合は以下のように定義する。

定義 5.3

$R \bowtie_{A_i \theta B_j} S$

$R \bowtie_{A_i \theta B_j} S = T \cup (C \times S')$

$T = A_i \theta B_i$

$S' = S - T[B_1, B_2, \cdots, B_m]$

$C(B_1, B_2, \cdots, B_m)$:全ての属性値が *null* 値の 1 タプルからなる定値リレーション

ただし、*R* $[A_1, A_2, \cdots, A_n]$, *S* $[B_1, B_2, \cdots, B_m]$

（4） S 表同士の結合は以下のように定義した。

定義 5.4

$S_a \bowtie_{A_i \theta B_j} S_b$

$S_a \bowtie_{A_i \theta B_j} S_b = T \cup (S'_a \times D) \cup (C \times S'_b)$

$T = A_i \theta B_j$

$S'_a = S_a - T[A_1, A_2, \cdots, A_n]$

$S'_b = S_b - T[B_1, B_2, \cdots, B_m]$

$C(B_1, B_2, \cdots, B_m)$:全ての属性値が *null* 値の 1 タプルからなる定値リレーション

$D(A_1, A_2, \cdots, A_n)$:全ての属性値が *null* 値の 1 タプルからなる定値リレーション

ただし、S_a $[A_1, A_2, \cdots, A_n]$, S_b $[B_1, B_2, \cdots, B_m]$

即ち SR モデルにおける θ-結合は、上記の (2)-(4) の場合、それぞれ LEFT OUTER JOIN、RIGHT OUTER JOIN、FULL OUTER JOIN に置き換えて演算が行われる。

5.3.3 R 表と S 表の結合演算の例

5.3.3 節のような R 表と S 表の JOIN の例を示すために、以下のようなサッカー日本代表データベースのスキーマの表を利

用する。

試合結果 (日付, 対戦相手国, スコア)
得点者 (日付, 選手名)
ここで、「得点者. 日付」は試合結果への外部キーである。

上記のスキーマにおいて、「久保選手が得点した試合の日付とその対戦相手国」を検索したいといった SQL クエリはクエリ 5.1 のようになる。

クエリ 5.1
久保選手が得点した試合の日付とその対戦相手国

```
SELECT 日付, 対戦相手国, 選手名
FROM 試合結果, 得点者
WHERE 試合結果. 日付 = 得点者. 日付,
        得点者. 選手名 = ' 久保'
```

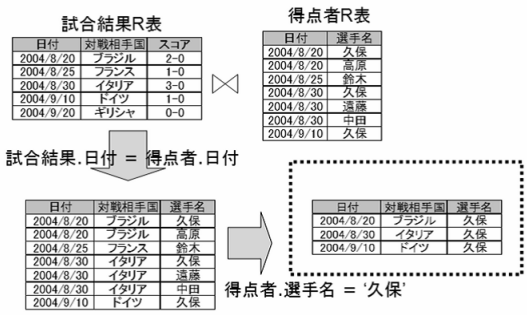

図 6　R 表同士の JOIN の例

クエリ 5.1 を実行した結果は図 6 のようになる。WHERE 句で指定された条件による選択の結果、もともとデータベースにはどの日に試合が行われたのか、という情報を含んでいたのにも関わらず、結果からは久保選手が 3 試合連続でゴールしているのか、それとも間にゴールできなかった試合が行われていたのかを読み取ることができない。そこで、SR モデルでは、5. 1 節のように定義することにより、行われた一連の試合の日付を結果に反映することができる。以下の SQL-SF クエリ 5.2 を実行した結果が、図 7 である。

クエリ 5.2
久保選手が得点した試合の日付とその対戦相手国

```
SELECT 日付, 対戦相手国, 選手名
FROM 試合結果, 得点者
WHERE 試合結果. 日付 = 得点者. 日付,
        得点者. 選手名 = ' 久保'
SEQUENCE ON 試合結果. 日付
```

SR モデルでは、結合演算を行う際に S 表の各タプルに対応するタプルがなくなってしまった場合でも、該当する属性には

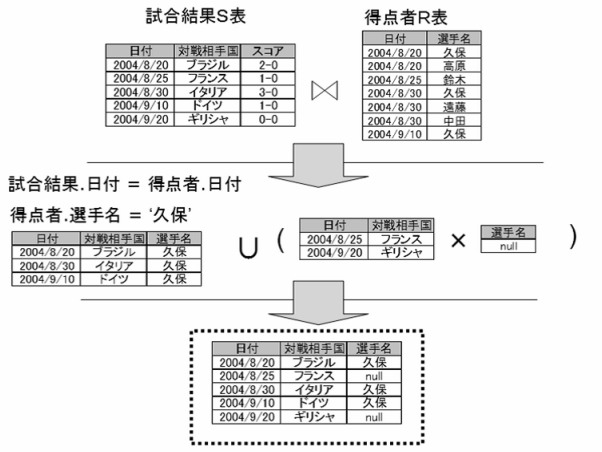

図 7　S 表と R 表の JOIN の例

null を設定し出力することにより、その連続性を失わないようにしている。(5. 3. 2 節) 図 7 の結果では、試合結果の S 表と得点者の R 表が「試合結果. 日付」、「得点者. 日付」の属性について結合されているが、このとき 2004/9/20 のギリシャ戦に対応する得点者タプルが存在しないため、出力結果の選手名属性には null が設定される。また、試合結果と得点者を結合した表には、「得点者. 選手名 = ' 久保'」の条件を満たすタプルが存在しないが、選択演算の後、試合結果の「2004/8/25」のタプルに対応するタプルが存在しなくなってしまうため、同様に選手名の属性には null を出力し、「2004/8/25」の試合結果のタプルを残している。このように、全ての試合に関するタプルを残すことで、試合結果の S 表の連続性を保存することが可能となっている。

6.　システム概要

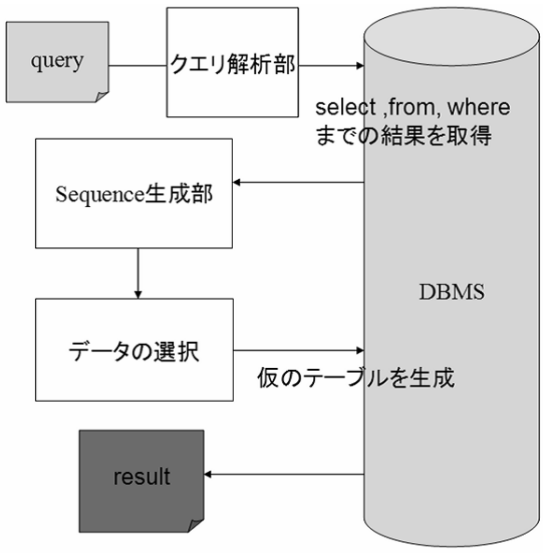

図 8　システム概要

3. 章で提案した言語仕様に則って、SR モデルに従った DBMS のプロトタイプを実装した。図 8 はそのシステムの概要である。

6.1 クエリ解析部

クエリ解析部では、クエリを SELECT、FROM、WHERE、SEQUENCE ON、ABOUT、SUCH、GROUP BY、HAVING、ORDER BY の各要素に分解する。FROM 句で複数の表が指定されている場合は、5.3.2 節におけるどの JOIN への書き換えを行うかを判断するため、どの表の属性に対し SEQUENCE ON 句が指定されており、どの表が S 表に変換されるのかを判断するのもこの部分である。

6.2 Sequence 生成部

6.1 節で解析された結果を用い、まず DB に対し SELECT、FROM、WHERE の部分までの問い合わせを行い結果を保存する。次に、その結果を SEQUENCE ON 句に従って S 表に変換し、ABOUT 句で指定された属性ごとに S 表をグループ化する。最後に SUCH 句の条件に従いデータの選択を行い、その結果を一時的な表として DB に格納する。

この部分が従来の SQL に対する SQL-SF の主となる拡張部分であるといえる。

6.3 結 果 出 力

最後に、GROUP BY、HAVING、ORDER BY の処理を行い、結果を出力する。

従来の SQL では、２次元のモデルであるリレーショナルモデルが採用されているため、GROUP BY が指定されている場合の出力結果はタプル-属性平面のグループ表のみであった。しかし、SR モデルでは、グループ化後もシーケンス-属性平面において各タプルが意味を持っているので、タプル-属性平面の出力の他に、シーケンス-属性平面での出力も可能でなくてはならない。そこで、出力形態の指定を SELECT 句によって行うことができるよう実装を行った。

以下のクエリは、ある Web サイト(注2) のアクセスログにおいて、4.2 節のスキーマを用いて検索を行った例である。

クエリ 6.1

today.html の次に閲覧されたページの集計

```
SELECT filename, count(*) FROM access
WHERE filename LIKE '%html'
        OR filename LIKE '%php'
SEQUENCE ON requestdate, requesttime
ABOUT hostname
SUCH PREV.filename
        = '/kawaguchi/nikki/today.html'
GROUP BY filename
ORDER BY count(*) DESC;
```

クエリ 6.1 において、SELECT 句で指定されている属性は、「filename」と「count(*)」であり、これは、GROUP BY で指定されている属性 (filename) と集計関数 (count(*)) であるため、出力結果は従来の SQL と同様、図 9 のようにタプル-属性平面のグループ表の出力になる。

(注2)：http://www.gkisland.com/kawaguchi/

filename	count
/kawaguchi/bbs/bbs.html	219
/kawaguchi/toppage/shiken.html	32
/kawaguchi/nikki/nikki.html	27
/kawaguchi/profile/profile.html	23
/kawaguchi/toppage/kawaguchimain.php	14
...	
/kawaguchi/others/othersmain.html	1

図 9　タプル-属性平面の出力

これに対し、以下のクエリ 6.2 では、SELECT 句に GROUP BY 句で指定されている「filename」属性以外に、「hostname」属性が指定されている。

クエリ 6.2

today.html の次に閲覧されたページのファイル名とアクセスしたホスト名

```
SELECT filename, hostname FROM access
WHERE filename LIKE '%html'
        OR filename LIKE '%php'
SEQUENCE ON dates
ABOUT hostname
SUCH PREV.filename
        = '/kawaguchi/nikki/today.html'
GROUP BY filename
```

従来の SQL では当然エラーが出るところであるが、SQL-SF では、シーケンス-属性平面での出力が可能であるため、図 10 のような出力結果になる。

filename	hostname
/kawaguchi/bbs/bbs.html	i220-108-252-169.s02.a013.ap.plala.or.jp
├/kawaguchi/bbs/bbs.html	h-68-164-115-229.isanca54.dynamic.covad.net
├/kawaguchi/bbs/bbs.html	od8.sharp.co.jp
...	
├/kawaguchi/bbs/bbs.html	ktcl55159.tmtv.ne.jp
├/kawaguchi/bbs/bbs.html	q053050.ppp.asahi-net.or.jp
├/kawaguchi/bbs/bbs.html	nttkyo064029.tkyo.nt.adsl.ppp.infoweb.ne.jp
├/kawaguchi/bbs/bbs.html	eq252.ade3.point.ne.jp
/kawaguchi/toppage/shiken.html	FLHIAbo235.osk.mesh.ad.jp
├/kawaguchi/toppage/shiken.html	61-26-139-12.rev.home.ne.jp
├/kawaguchi/toppage/shiken.html	219-122-202-35.eonet.ne.jp
├/kawaguchi/toppage/shiken.html	i218-47-20-1.s02.a013.ap.plala.or.jp
...	
/kawaguchi/others/othersmain.html	p5006-ipad33sapodori.hokkaido.ocn.ne.jp

図 10　シーケンス-属性平面の出力

シーケンス-属性平面において各タプルごとの出力を行う場合でも、GROUP BY で指定されたグループは「├–」を用いて表現している。クエリ 6.2 では、ファイル名ごとにグループ化しているが、図 10 のように、ファイル名のグループごとに表示をしている。

このように、結果の出力形態は、SELECT 句で指定された属性により異なる。尚、出力がタプル-属性平面になるのは、SELECT 句が、

（1） GROUP BY 句で指定された属性
（2） 集計関数
（3） FIRST, LAST の二つの特殊タプル変数

のみで構成されている場合である。それ以外の属性が一つでも SELECT 句に含まれている場合、あるいは SELECT 句で*が指定されている場合はシーケンス-属性平面での出力になる。

6.4 システム評価

本システムでは、S 表を生成した後にデータを仮想的な表として再び DB に格納しているため、その INSERT 処理が大きなボトルネックとなってしまい、処理速度において、DBMS としての実用性があるとはいい難い。検索速度を改善するような実装や、検索の最適化は今後の課題であるといえる。

AQuery [4] では、集計関数として、mins() や maxs() などを実装している。これは、現在評価しているタプル以前までで最小・最大の値を得る関数であり、連続性にもとづいた検索特有の集計関数である。このような関数も、SR モデルに従えば実装可能である。また、現在のところ 3.3 節で示した４つの特殊タプル変数にしか対応していないが、PREV(n)、NEXT(n) のように、n 個前、n 個先のタプルを示す特殊タプル変数の実装なども検討中である。グルーピングキーワードについても、現在は CONSECUTIVES のみ定義されているが、順序が定義されたことで、タプルを n 個ずつグループ化していくなど他にもいくつかのグルーピングキーワードが追加可能であると考える。

GROUP BY 句が指定されているときの出力形態については、6.3 節で述べたような実装を行っている。しかし、これをモデル化するには SR モデルにおける射影演算を拡張定義しなおす必要がある。このように、関係演算の再定義も今後の課題であるといえる。

7. ま と め

本稿では、データの連続性にもとづいた検索クエリを記述可能にする検索言語 SQL-SF を提案した。また、その基盤となるデータモデルである、SR モデルをリレーショナルモデルの純粋な拡張として示した。SQL-SF は SR モデルに従って拡張させることが可能であり、より有用なグルーピングキーワードや特殊変数を取り入れていくことができる。今後は、SR モデルにおける関係代数の定義を行うと共に、検索の最適化を行うことにより、DBMS としてより実用的なものにしていくことを考えている。

文　　献

[1] Arunprasad P. Marathe and Kenneth Salem. ”Query Processing Techniques for Arrays.” *The VLDB Journal*, 11: pp68-91, 2002

[2] Knuth, D., Morris, J. , and Pratt, V. ”Fast Pattern Matching in Strings.” *Journal of SIAM on computing* June 1977, pp323-350

[3] KX Systems. ”KSQL Reference Manual, Version 2.0” http://www.kx.com

[4] Lerner, A., and Shasha D., ”AQuery: Query Language for Ordered Data, Optimization Techniques, and Experiments”, *Proceedings of the 29th VLDB Conference, Berlin*, pp.345-356, 2003

[5] Libkin, L., Machlin, R., and Wong, L., ”A Query Language for Multidimensional Arrays: Design, Implementation, and Optimization Techniques.” *SIGMOD Int’l Conf on Management of Data*, pp228-239 1996

[6] Melton, J., ”Advanced SQL:1999 - Understanding Object-Relational and Other Advanced Features.” Morgan Kaufmann Publishers, 2002.

[7] Ramakrishnan, R., Donjerkovic, D., Ranganathan, A., Beyer, K.S., and Krishnaprasad, K., ”SRQL: Sorted Relational Query Language.”, *SSDBM Int’l Conf on Scientific and Statistical Database Management*, pp84-95, 1998

[8] Sadri, R., Zaniolo, C., Zarkesh, A., and Adibi, Jafar, ”Expressing and Optimizing Sequence Queries in Database Systems” *ACM Transactions on Database Systems*, Vol.29, No.2, June 2004, pp 282-318

[9] Seshadri, P., Livny, M., and Ramakrishnan, R. ”The Design and Implementation of a Sequence Database System.” *VLDB Int’l Conf on Very Large Data Bases*, pp99-110, 1996