

SQL 言語に対するハッシュ操作演算系での実現方式

3N-8

中山雅哉

斎藤制海

(豊橋技術科学大学, 知識情報工学系)

1. はじめに

近年、関係データモデルに基づくデータベースシステムの構築が盛んに行なわれており、商用システムを含むこれらのデータベースシステムの多くでは、ISO で標準化された SQL 言語をユーザ問合せ言語として採用している。しかし、ISO 規格を完全にサポートするシステムは少なく、特に繰り返し構造を持つ問合せ式 (Nested Query) に関する記述に制限を加えるものが多い。これは、SQL 言語の持つ記述形式の複雑さに対して制限が加えられているというよりも、むしろ繰り返し構造を持った問合せ式に対する安易な実現方式に基づく処理性能の限界から生ずる制約であると考えられる。

しかしこの問題は、文献 [2,3] 等で述べられているように、繰り返し構造を持った問合せ式が、複数個の繰り返し構造を持たない問合せ式で等価に表現できることを利用すれば、SQL 言語の基本問合せ式 (繰り返し構造を持たない SELECT—FROM—WHERE—GROUP-BY—HAVING 構文をとる問合せ式を“基本問合せ式”と呼ぶ) に対する高速な演算処理方式と複数の問合せ式に対する一貫性制約を保障するトランザクション処理手法を用意することで解決することができる。

このうち、応用言語からのデータベース利用において必要不可欠なトランザクション処理については他の文献に譲り [1]、本稿ではハッシュ操作に基づく演算処理手法による SQL 問合せ式の実現方式についてまとめている。

以下、2 章では SQL 言語の文法を概説し、一般問合せ式が 2 種類の基本問合せ式の組み合わせで表現できることを示す。

3 章では、各基本問合せ式が我々の提案するハッシュ操作に基づく演算処理方式と親和性が高く高速に演算処理を施すことができることを示すとともに、一般問合せ式を木構造で表現した際の、基本問合せ式間の処理効率を向上させる方法について述べる。

4 章では本稿のまとめを行なうとともに、複数プロセッサ構成の並列データベースシステム上での実現方式への適用性について検討を行なっている。

2. SQL 言語の一般形式と基本問合せ式の分類

文献 [4] 等々に示されているように、SQL 言語では以下のような構文が許されている。

問合せ式 ::= 問合せ項 | 問合せ式 UNION [ALL] 問合せ項
問合せ項 ::= 問合せ指定 | (問合せ式)

問合せ指定 ::=

```
SELECT [ALL | DISTINCT] {スカラ式コンマリスト | *}
FROM 表参照コンマリスト
[ WHERE 探索条件 ]
[ GROUP BY 列参照コンマリスト ]
[ HAVING 探索条件 ]
```

ここで、[] は省略可能であることを示し、{ } は必須なパラメタの選択指定であることを示す。そして、| は選択子の区切りを示している。

```
P (pnum, qoh)
S (pnum, quan, shipdate)
```

```
SELECT pnum
FROM P
WHERE qoh = ( SELECT COUNT(shipdate)
               FROM S
               WHERE S.pnum = P.pnum
               AND shipdate < 1-1-80 )
```

図 1: 繰り返し構造を持った問合せ式の例

このように一般の問合せ式は“問合せ指定”を組み合わせで表現することができるが、“問合せ指定”自身が‘探索条件’内で利用できるため、繰り返し構造を持った形で表現されることがある (図 1)。

探索条件 ::= ブール項 | 探索条件 OR ブール項
ブール項 ::= ブール因子 | ブール項 AND ブール因子
ブール因子 ::= [NOT] {述語 | (探索条件)}

述語 ::=

```
スカラ式 比較 {スカラ式 | 副問合せ}
| スカラ式 [NOT] BETWEEN スカラ式 AND スカラ式
| 列参照 [NOT] LIKE アトム [ESCAPE アトム]
| 列参照 IS [NOT] NULL
| スカラ式 [NOT] IN {副問合せ式 | アトムコンマリスト}
| スカラ式 比較 [ALL | ANY | SOME] 副問合せ
| EXISTS 副問合せ
```

比較 ::= = | < | <= | > | >=

副問合せ ::= (問合せ指定)

従来のデータベースシステムではこのような繰り返し構造を持った問合せ式は、単純に内側の“問合せ指定”から順に実行する方法がとられてきた。図 1 に示すような階層構造をまたがった結合演算処理を含む問合せに対しては、外側のリレーションから各タプルを取り出して値を代入し、結合演算を選択演算に変形しながら内側の“問合せ指定”を実行する方法がとられる。この方法では、外側のリレーションのタプル数分だけ内側の処理を実行する必要がある、大規模リレーションを対象とする場合には実行時間での演算処理が望めなくなる。

これに対して、文献 [2,3] に示されている様に、繰り返し構造を持つ問合せ式を複数個の繰り返し構造を含まない問合せ式 (基本問合せ式) に変換する手法を用いれば、内側処理を反復して実行する必要がなくなる。例えば、図 1 に示した問合せ式は、以下に示す 2 つの基本問合せ式により表現できる。

```
TMP(supnum, ct) =
( SELECT P.pnum, COUNT(S.shipdate)
  FROM P, S
  WHERE P.pnum = S.pnum
  AND S.shipdate < 1-1-80
  GROUP BY P.pnum )
```

```

SELECT P.pnum
FROM   P, TMP
WHERE  P.qoh = TMP.ct
AND    P.pnum = TMP.supnum

```

ここで、“=”は、外部結合演算を表している。

さらに、SQL 言語で記述された問合せ式は、

“FROM 句で示したリレーシンの各タプルから WHERE 句で指定した条件を満足するものだけを抜き出し、GROUP-BY 句で示した属性群でグループ化した後、HAVING 句で示される条件を満足するグループを抜き出して結果として出力する”

という順序で実行されることが定められているため、上の例で TMP を出力する基本問合せ式は、次の 2 つの基本問合せ式を用いて記述できることになる。

- (1) 複数のリレーシオンに条件を与えて 1 つにまとめる基本問合せ式
- (2) 特定の属性でグループ化した際の条件評価を行なう基本問合せ式

```

TMP1(pn, sd) =
( SELECT P.pnum, S.shipdate FROM P, S
  WHERE P.pnum =+ S.pnum
    AND  S.shipdate < 1-1-80 )

```

```

TMP(supnum, ct) =
( SELECT pn, COUNT(sd) FROM TMP1 GROUP BY pn )

```

これらの基本問合せ式は、次章に示すようにハッシュ操作に基づいた演算処理手法と親和性が高く、大規模リレーシオンに対しても高速に演算処理を行なうことが可能となる。

3. ハッシュ操作演算系での実現方式

2 種類の基本問合せ式のうち、(1) 結合演算処理を含むものは、文献 [5] 等で示したようにハッシュ操作に基づいた演算処理手法を用いることができる。この手法は、特に大規模なリレーシオンに対して有効に作用する。

また、(2) GROUP-BY 句に対する処理も、指定された属性値に従ってハッシュ操作を施し、同一のハッシュ値を有するバケット内で集約演算処理等を行ないながら射影・選択処理を施すことに相当し、ハッシュ操作を用いることで自然に演算処理を実現することが可能となる。

Rel = (group, other)

```

SELECT R.a, S.a
FROM   R, S
WHERE  R.b = S.b
AND    R.c & S.c
AND    R.d & R_const
AND    S.d & S_const
SELECT group, AGGR_S(other)
FROM   Rel
WHERE  Rel.d & Rel_const
GROUP BY group
HAVING AGGR_H(other) & 0_const

```

(1) 結合タイプ

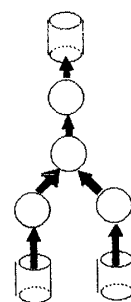
(2) GROUP-BY タイプ

(注) & は任意の比較演算を & ' は等号以外の比較演算をさす

ここで、各タイプの基本問合せ式を一般的に表すと上式のようになり、問合せ式中の各操作を選択演算型操作と結合演算型操作に分類することで図 2 に示すような実行木が得られる。

このように各基本問合せ式は、(複数の) 入力リレーシオンから出力リレーシオンを生成する一連の流れとして定義できる。

前章で述べたように一般の問合せ式は、基本問合せ式のまとまりとして表現することができるため、その実行木は基本問合せ式の入出力関係を結びつけることで図 3 に示すように表すことができ、選択演算型の操作が続いて現れる場合は、あわせて



(1) 結合タイプ



(2) GROUP-BY タイプ

図 2: 各基本問合せ式に対する実行木

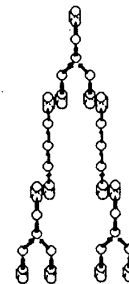


図 3: 一般問合せ式に対する実行木の例

処理を行なうようにすることで余分な入出力コストを軽減することが可能となる。

4. まとめ

本稿では、SQL 言語で記述される一般問合せ式を、繰り返し構造を持たない基本問合せ式の組合せとして表すことで、ハッシュ操作に基づいた演算処理系を用いて容易に実行できることを示してきた。

ここで示した一般問合せ式に対する実行木を単一プロセッサ計算機上に実装する場合は、ハッシュ操作に伴う主記憶領域の不足とタスク切り換えによる処理オーバーヘッドの増加から、各々の演算処理を中間リレーシオンを用いて接続する方式をとる必要があるが、複数プロセッサ構成の並列計算機上で実装する場合は、各プロセッサに各演算処理を割り当てることで中間リレーシオンを介さない方式をとることができる。

この演算処理システムについては、現在試作システムの構築を進めており、詳細については別稿で発表する予定である。

参考文献

- [1] J. D. Ullman. "Principles of Database Systems". Computer Science Press, second edition, 1982.
- [2] W. Kim. "Query Optimization for Relational Data Base Systems". In P. S. Fisher, J. Slonim, and E. A. Unger, editors, *Advances in Data Base Management, Vol. 2*, chapter 9, pp. 171-195. Wiley Heyden Ltd., 1983.
- [3] R. A. Ganski and H. K. T. Wong. "Optimization of Nested SQL Queries Revisited". In *ACM SIGMOD '87*, pp. 23-33, 1987.
- [4] C. J. Date, 芝野耕司 監訳, 岸本令子 訳. "標準 SQL (A Guide to THE SQL STANDARD)". トッパン, 1988.
- [5] M. Nakayama, M. Kitsuregawa, and M. Takagi. "Hash-Partitioned Join Method Using Dynamic Destaging Strategy". In *Proc. of the 14th Int. Conf. on VLDB*, pp. 468-478, 1988.