Vol. 34 No. 7 情報処理学会論文誌 July 1993

ショートノート

SQL 質問における基数検査について

佐藤 隆士†

データベース言語 SQL は，データベースの標準インタフェースとして定着している．SQL 質問の探索条件のうち，比較述語内に副問い合わせが現れる場合，その結果は1列で，かつたかだか1行の表でなければならない．このための検査を基数検査という．本論文では，質問の最適化を行った場合の基数検査の方法を提案する．

On Cardinality Checks of SQL Queries

TAKASHI SATO†

SQL is a database language attracting the attention of many people for standard database interface. When a subquery appears in a comparison predicate, its result table should be one column by at most one row. The check of this restriction is called cardinality check. The author proposes a way of cardinality check when queries are optimized.

1. はじめに

SQL は，ブロック構造をもつデータベース言語で，各種機関で規格化が進み，データベースの標準インタフェースとして定着している．SQL の問い合わせ文は，SELECT 句，FROM 句，WHERE 句などからなるが，このうち WHERE 句の WHERE 以下に続く条件を探索条件という．探索条件に比較対象が副問い合わせになっている比較述語が現れる場合，その結果は1列でかつたかだか1行の表でなければならない[1]．列数は副問い合わせの SELECT 句からすぐわかるが，行数については主キーに対する検索条件が含まれていない限り，実際のデータ（関係表）を調べなければ判断できない．このための検査を基数検査という．副問い合わせをもつ質問は，入れ子質問と呼ばれる．入れ子質問の処理は，コストが大きくなることが多いので，実用的なシステムでは最適化を行う必要がある．最適化を行う場合，従来の手法では，基数検査ができなくなってしまうので注意が必要である．そこで，本論文では，最適化を行いかつ基数検査もできる方法を提案する．

† 大阪教育大学教養学科
Department of Arts and Sciences, Osaka Kyoiku University

2. 用語の説明

関係データベースは，論理的には関係と呼ばれる表形式のデータの集まりである．表の行数のことを基数という．表を操作する代表的な演算子として選択，射影，結合がある．このうち結合は，二つの表を R_a, R_b とするとき，R_a のある列 C_a と R_b の列 C_b の値が指定された条件を満たす両表の行を連接した行からなる新しい表を作る演算である．データベース言語 SQL の質問ブロックは，SELECT 句，FROM 句，WHERE 句などから構成されるが，WHERE 句の中にさらに質問ブロックを含むことができる．すなわち入れ子構造をなすことができる．2重の入れ子構造について二つの質問ブロックをそれぞれ外側および内側と呼ぶ．外側の WHERE 句に比較述語（$=, \neq, <, >, \leqq, \geqq$ のうちのどれか）が現れる場合，内側の質問ブロックの結果は1列でかつたかだか1行の表でなければエラーである[1]．このうち行数の検査を基数検査という．

3. 副問い合わせをもつ質問と最適化

Kim[2] は2重の入れ子構造をもつ質問を五つのタイプに分類し，入れ子構造を含まない質問に変換する方法と，これら変換の組み合わせにより，任意の入れ子構造をもつ質問を，入れ子を含まない形に変換できる

ことを示した．現在の SQL 規格では，2重の入れ子構造は，(1)内側の質問ブロック中に外側の質問ブロックの表の参照（外への参照という）の有無，(2)内側の SELECT 句中に集合関数の有無，の組み合わせで**表1**に示す四つのタイプが可能である．

これらのうち，タイプNは，内側の質問ブロックが外側の質問ブロックと独立に処理できるので，最適化後も基数検査は容易である．タイプAとタイプ JA では，副問い合わせの SELECT 句には，列数が1の制約から，集合関数を用いた1列が現れる場合のみを考えればよいので，結果が2行以上になることはなく，基数の条件は常に満足される．したがって，問題はタイプ J の場合に絞られる．実際，Kim[2] の最適化手法のままでは基数検査を行わないので不都合が生じる．以下では，理解を容易にするため，具体的に二つの関係スキーマ

PARTS(PNUM, PNAME, QOH)—部品関係
SUPPLY(PNUM, QUAN, SHIPDATE) —供給関係

に対する，次に示すタイプ J の SQL 文について議論することにする．

質問 Q1:

```
SELECT PNUM
FROM PARTS
WHERE QOH=(SELECT QUAN
  FROM SUPPLY
  WHERE SUPPLY.PNUM=PARTS.PNUM
  AND SHIPDATE<1-1-80);
```

「1-1-80 以前のある供給量 (QUAN) と同じ在庫量 (QOH) をもつ部品の部品名 (PNAME) を答えよ」という質問である．内側のブロックにある PARTS.PNUM が外への参照になっている．SUPPLY については，PNUM は主キーではないので，副問い合わせの基数は1を超える可能性がある．

4. 基数検査の最適化

質問 Q1 を最適化せずにそのまま実行すると，入れ子反復 (nest iteration) という，二つの表のすべての行の組み合わせについて突合わせを要する処理になる．すなわち，PARTS 表の行数を m とすると m 回，SUPPLY 表のうち SHIPDATE<1-1-80 の選択条件に合うもの（行数 n とする）を二次記憶から主記憶に読み込むことになる．本論文では，処理コストをページ読込回数で見積もることにし，表の占めるページ数をそれぞれ P_m, P_n とすると，この際のページフェッチ数は，

$$C_N = mP_n \qquad (1)$$

となる．m, n が大きい場合，C_N は非常に大きくなるのでこの方法は実用的ではない．

表 1 二重の入れ子構造の四つのタイプ
Table 1 4 types of nested query.

		outer reference	
		no	yes
aggregate function	no	type N	type J
	yes	type A	type JA

4.1 質問の入れ子構造解消による最適化

質問 Q1 の内側のブロックについて，

```
TEMP1(CNT)=SELECT COUNT(*)
    FROM SUPPLY
    WHERE SHIPDATE<1-1-80
    GROUP BY PNUM;
```

とし，TEMP1 の CNT 列の中に，1を超える行があれば基数検査でエラーといってもよさそうに見える．しかし，その行が由来する，SUPPLY.PNUM が PARTS.PNUM の中になければエラーは出さないようにしなければならない．つまり，SUPPLY 表だけでは判定できない．外への参照をもつ場合は，本質的に表の結合を要する．この例では，SUPPLY と PARTS の関係の結合が必要である．

```
TEMP2(PNAME, QOH, QUAN, PPNUM)=
    SELECT PNAME, QOH, QUAN,
      PARTS.PNUM
    FROM PARTS, SUPPLY
    WHERE PARTS.PNUM=SUPPLY.
      PNUM
    AND SHIPDATE<1-1-80;
TEMP3(CNT)=SELECT COUNT(*)
    FROM TEMP2
    GROUP BY PPNUM;
```

とし，TEMP3 の CNT 列の中に1を超える行があれば基数エラーと判定する．基数エラーでなければ，

```
SELECT PNAME
FROM TEMP2
WHERE QOH=QUAN
```

で結果を得る．

まず，TEMP2 作成の際のページ読込みコストを，代表的な結合処理方式である，(1)入れ子ブロック結

合，(2)併合結合，でそれぞれ結合処理を行った場合について見積もる．

(1) 入れ子ブロック結合[3)]

結合処理で使用可能な主記憶のデータページ数を B とする．$B-1$ ページ分に PARTS 表を読み込み，主記憶の残り1ページに選択条件に合う SUPPLY 表の部分を順次読み込みながら結合を行うと，

$$C_B=P_m+\lceil P_m/(B-1)\rceil P_n \qquad (2)$$

となる．

(2) 併合結合

併合結合の場合は，表をあらかじめ結合列でソートし，両表の先頭から走査し，結合条件を満たす行を連接する．コストは，

$$C_M=P_n\lceil\log_B P_n\rceil+P_m\lceil\log_B P_m\rceil+P_n+P_m \qquad (3)$$

となる．

TEMP2 の大きさを P_j ページとすると，基数検査のための TEMP3 の作成および，その後の処理のコストは，

$$P_j\lceil\log_B P_j\rceil+P_j+P_j \qquad (4)$$

となる．ただし，上式第1項は GROUP BY の際のソートのコストだが，併合結合の場合はソート済みなので不要である．したがって，全体ではそれぞれ，

$$C_{B+}=P_m+\lceil P_m/(B-1)\rceil P_n+P_j\lceil\log_B P_j\rceil+2P_j \qquad (5)$$

$$C_{M+}=P_n(\lceil\log_B P_n\rceil+1)+P_m(\lceil\log_B P_m\rceil+1)+2P_j \qquad (6)$$

となる．

4.2 基数検査オプション付きの結合

4.1 節では結合と基数検査を別々に行ったが，結合を行う際，基数検査も並行して行うと，さらに性能を改善できる可能性がある．すなわち，結合処理中に基数条件を監視することにより，基数違反が検出された時点で基数エラーを報告し，残りの結合処理を取りやめることができる．具体的には，質問 Q1 を

```
SELECT PNAME
FROM   PARTS, SUPPLY
(1) WHERE PARTS.QOH=SUPPLY.
          QUAN
(2) AND   PARTS.PNUM=SUPPLY.
          PNUM
(3) AND   SHIPDATE<1-1-80;
```

と変換する．ここで，(1)，(2)，(3)は説明のための WHERE 句の条件番号である．これは，Kim[2)] の変換アルゴリズム (NEST-N-J) によるものであり，このまま普通に処理したのでは，基数検査は正しくできない．そこで処理方法を工夫することにより基数検査を行うことができるようにする．すなわち，最初に(3)で SUPPLY 表から条件に合う行を取り出し，次に(1)と(2)の条件で結合を行う．この際，まず(2)の条件で PARTS.PNUM に一致する SUPPLY.PNUM が二つ以上発見されたとき基数エラーを報告し処理を中断する．基数エラーが見つからない限り(1)の条件も適用し結合処理を進める．(1)の条件での結合処理の直前に，(2)の処理で基数検査を行う点が特徴である．

以上の方法によりどの程度改善が期待できるか見積もる．4.1 節に比べ，基数違反がない場合でも，中間結果 TEMP2 の読込み $2P_j$ ページフェッチが減少できる．基数違反がある場合，結合処理中断によるコストの軽減を見積もる．延べ a 回ランダムに基数違反が生じると仮定する．入れ子ブロック結合では，式(2)に対して，$1/(a+1)$ の $C_B/(a+1)$ ページフェッチ分処理した時点で基数エラーを報告し終了するので，処理コストは大幅に改善される．一方，併合結合では，ソートの部分のコストは軽減されないため，式(3)のうち併合部のコスト P_m+P_n が $1/(a+1)$ に減少される．したがって，基数検査を行いながら結合処理を行うことにより(5)，(6)はそれぞれ，

$$C'_{B+}=(P_m+\lceil P_m/(B-1)\rceil P_n)/(a+1) \qquad (7)$$

$$C'_{M+}=P_n(\lceil\log_B P_n\rceil+1/(a+1))+P_m(\lceil\log_B P_m\rceil+1/(a+1)) \qquad (8)$$

表 2 コストの比較例
Table 2 Example of processing costs.

P_m	C_N	C_{B+}	C'_{B+}				C_{M+}	C'_{M+}			
			$a=0$	3	10	100		$a=0$	3	10	100
30	9000	180	150	38	14	1	200	180	135	125	121
100	100000	1330	1300	325	118	13	620	600	450	418	402
300	900000	10530	10500	2625	955	104	2420	2400	1950	1855	1806
1000	10000000	113030	113000	28250	10273	1119	8020	8000	6500	6182	6020

に改善される．TEMP2に相当する中間結果は作成しないので，P_jの項は現れない．このように基数検査オプションを導入することにより処理コストをさらに減少することができる．

（例1）$m=n=10P_m$, $P_n=P_m$, $P_j=10$, $B=10$として，$P_m=30, 100, 300, 1000$, $a=0, 3, 10, 100$について，式(1)，(5)，(6)，(7)，(8)からC_N, C_{B+}, C'_{B+}, C_{M+}, C'_{M+}を求めた結果を**表2**に示す．一見して，C_Nに比べ他は大幅にコストが減少していることが明らかである．C'_{B+}, C'_{M+}における$a=0$は，基数条件を満足する場合でありそれぞれC_{B+}, C_{M+}と大差はないが，aが大きくなるにつれさらにコストが減少する様子がよくわかる．とくに入れ子ブロック結合の場合は，より効果的であることが確認できる． □

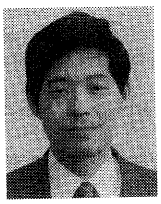
5. おわりに

本論文では，基数検査を行う必要がある副問い合わせをもつSQL質問の最適化について具体例で論じ，最適化を行わない場合に比べ処理コストを大幅に低減できることを示した．また，結合処理に基数検査を含ませることにより，中間結果の読込みを不要にし，かつ基数違反が発見され次第処理を打ち切ることができるので，さらに効率化できる．結合処理が入れ子ブロック結合の場合は，特に効果が大きいことを示した．

参考文献

1) Date, C. J.（芝野耕司監訳）：標準 SQL-改訂第2版，トッパン，p. 137 (1990)．
2) Kim, W.: On Optimizing an SQL-like Nested Query, *ACM Trans. Database Syst.*, Vol. 7, No. 3, pp. 443-469 (1982)．
3) Ullman, J. D.（國井ほか訳）：データベース・システムの原理，日本コンピュータ協会，p. 327 (1985)．

（平成4年2月6日受付）
（平成5年5月12日採録）



佐藤 隆士（正会員）

昭和28年9月生．昭和51年岡山大学工学部電子工学科卒業．昭和53年同大学院修士課程修了．同年詫間電波工業高等専門学校助手．現在大阪教育大学教養学科助教授．工学博士．データベース，記憶階層のアルゴリズム，並列処理の研究に従事．電子情報通信学会，CAI 学会，IEEE Computer Society, ACM 各会員．