データ・ベースシステム 71－5
（１９８９．５．１８）

ＳＱＬ３におけるカーソル／ビュー更新規則

麦谷　尊雄
日本電気株式会社　基本ソフトウェア開発本部・データベース開発部

寺田　克則
株式会社東芝　青梅工場　コンピュータ開発部

ＳＱＬは，１９８７年に国際標準化機構（ＩＳＯ）および日本規格協会において標準化された，関係データベースシステムのためのデータベース言語である。現在ＩＳＯでは，データベース言語ＳＱＬに対する拡張提案としてＳＱＬ２およびＳＱＬ３の国際標準化が進められている。本稿では，ＳＱＬ３の拡張機能として日本より提案したカーソル／ビュー更新規則について述べる。本提案は，ＳＱＬおよびＳＱＬ２で強く制限されているカーソル／ビューの更新規則を緩和するものである。

Cursor／View Updation Rule For SQL3

Takao Bakuya
Database Development Department,
Basic Software Development Division, NEC Corporation

Katsunori Terada
Computer Development Division,
TOSHIBA Corporation

SQL is a standard database language for the relational database systems published internationally by ISO (International Organization for Standardization) and domestically by JSA (Japan Standard Association) both in 1987. Two new languages to enhance the Database Language SQL, namely SQL2 and SQL3, are currently under progress for standadization by ISO. This paper describes the cursor/view updation rule which JAPAN proposed as an enhanced feature for SQL3. This new updation rule relaxes the severe restriction placed on the updatablility of the cursors and views in SQL and SQL2.

1．はじめに

ＳＱＬは，１９８７年に国際標準化機構（ＩＳＯ）および日本規格協会において標準化された，関係データベースシステムのためのデータベース言語である。現在ＩＳＯでは，データベース言語ＳＱＬに対する拡張提案としてＳＱＬ２およびＳＱＬ３の国際標準化が進められている。本稿では，ＳＱＬ３の拡張機能として日本より提案したカーソル／ビュー更新規則について述べる。本提案は，ＳＱＬおよびＳＱＬ２で強く制限されているカーソル／ビューの更新規則を緩和する。

以下に，本稿で用いる概念について記述する。

導出表は，＜問合せ指定＞を実行した結果，データベース内の一つ以上の表から導出される表であり，その値は，もとの表の値から導出される。

ビューは，＜ビュー定義＞(CREATE VIEW) によって定義される名前付きの導出表であり，データベース中の永続対象となる。ビューを定義する目的は，実表として定義した関係以外の関係を利用者の視点から定義し，それに対する検索や更新を可能とすることにより，利用者インタフェースを簡易化するとともに，機密保護のための機能を提供するものである。

カーソルは，＜カーソル宣言＞ (DECLARE CURSOR) により定義され，問合せ式で指定される導出表と，その表の行の順序およびその順序の相対位置を示す。＜カーソル宣言＞で指定される導出表は，＜ＯＰＥＮ文＞の実行により実効的に作成され，＜ＣＬＯＳＥ文＞の実行により実効的に破壊される。

このようなビューあるいはカーソルといった導出表に対する更新は，実際にはそれを導出するもととなった実表に対する更新に展開される。したがって，導出表に対する更新は，実表に対する更新が”一意に決定できる”場合に限られる。

2．ＳＱＬ，ＳＱＬ２における更新規則

ＳＱＬにおいてビューは，次のすべての規則が成り立つ場合に更新可能である。

(1) ＤＩＳＴＩＮＣＴ指定がない

(2) 選択リスト中のいずれの項目（値式）も１つの列指定から成り，どの列指定も２度以上現れない

(3) ＦＲＯＭ句がただ１つの実表，または更新可能な導出表を参照する（２表以上指定していない）

(4) ＷＨＥＲＥ句は，副問合せを含まない

(5) ＧＲＯＵＰ　ＢＹ句，ＨＡＶＩＮＧ句を含まない

カーソルは，ビューでの規則に加えて次の規則が成り立つときに更新可能である。

(6) ＯＲＤＥＲ　ＢＹ句が指定されていない

(7) 集合演算（ＵＮＩＯＮ）が指定されていない

ＳＱＬ２では，ＳＱＬに対する制限のうち，(4) が次のように緩和されている。

(4)’ＷＨＥＲＥ句は，更新の対象となる表を参照する副問合せを含まない。

このように，ＳＱＬおよびＳＱＬ２における更新規則は，非常に強く制限されている。なお，(6) と (7) の制限がカーソルのみに適用されるのは，ＳＱＬではビューにはＯＲＤＥＲ　ＢＹ句とＵＮＩＯＮの指定ができないからである（ＳＱＬ２以降では，ＵＮＩＯＮの他にＩＮＴＥＲＳＥＣＴ，ＥＸＣＥＰＴといった集合演算がビュー定義でも指定可能となる）。

上記の(1) ～ (7)の制限は，それぞれ次の様な問題を回避している。

(1) 更新が実表の重複した行に対するものとなる場合，実表の１つの行のみを更新するのか，それとも重複した行をすべて更新するのか，また１つの行のみを更新するならば，どの行を更新するのかが不明確である。

(2) 選択リストが集合関数を含む場合には実表に対する更新値が不明確である。また，２つ以上の列による演算式を含む場合には個々の列に対する更新値が不明確である。

(3) 複数の表の結合からなるカーソル／ビューに対する更新において，実表を更新した場合に，カーソル／ビューの他の行に対して意図しない結果が反映される可能性がある。

(4) 更新される実表が副問合せの中に指定されていると，意図しない結果が反映される場合がある。

(5) ＤＩＳＴＩＮＣＴと同様の問題がある。

(6) ＯＲＤＥＲ　ＢＹ句で指定された列の値が更新されるとその影響が不明確である。

(7) ＵＮＩＯＮを含むカーソルに対する更新では，それに伴う実表の更新が何通りか存在する場合があり，一意に決定できない。

3．ＳＱＬ３における更新ルール

3．1　基本的考え方

上記で述べたように，導出表に対する更新が許されるのは，それに伴う実表への更新が一意に決定できる場合に限られる。”更新が一意に決定できる”とは，以下の条件を満足する状態を言う：

(1) 更新の対象（表，行，列）が確定できる。

(2) 更新の種類が曖昧でない。

つまり，導出表に対する更新を実表への更新に変換するにあたり，どの表の，どの行を，どのように更新すべきであるが明確であり，かつ，そのような更新が１つ以上存在しない場合に限られる。

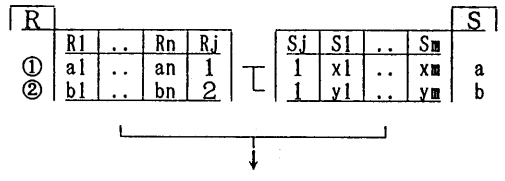
実際に更新することが可能である導出表の範囲は，導出表の実装方式によってＤＢＭＳごとに異なるが，ここでは，表に格納されている行を識別するために殆どのＤＢＭＳが内部的に持っているタプルＩＤ（以下ＴＩＤと称する）に着目し，導出表に対する付加情報としてＴＩＤを保持することによって，実表への更新を一意に決定できる範囲の導出表を更新可能とする。

上記の制限に加え，ＳＱＬでは，カーソル／ビューが更新可能であるか読み込み専用であるかは，更新命令実行時のインスタンスとは独立して，定義情報から判定可能でなければならない。

３．２　結合表の更新

複数の表を結合することによって導出される導出表については，結合結果をＴＩＤの組で保持することができる。結果が１対１となる結合であれば（同じＴＩＤが重複して現れなければ），その導出表は，更新可能である。結果が１対ｎとなる結合であっても，ｎ側の表（ＴＩＤが重複しない側）に限り更新可能である。

以下に示すように，１対ｎの結合において，１側の表を更新した場合，導出表に現れる他の行の値も更新されてしまうことになるため，このような更新は制限しなければならない。結果がｎ対ｎとなる結合についても同様である。




```
CREATE  VIEW V1 AS
  SELECT * FROM R,S WHERE Rj=Sj
```

V1

	R1	..	Rn	Rj	Sj	S1	..	Sm	
①	a1	..	an	1	1	x1	..	xm	a
①	a1	..	an	1	1	y1	..	ym	b

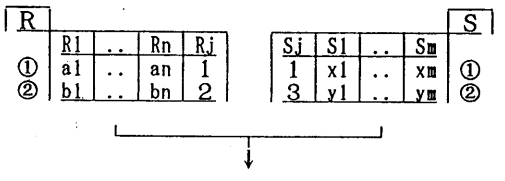
```
UPDATE V1 SET R1=w1 WHERE S1=x1
```

V1

	R1	..	Rn	Rj	Sj	S1	..	Sm	
①	w1	..	an	1	1	x1	..	xm	①
①	w1	..	an	1	1	y1	..	ym	②

上の例では，ビュー表Ｖ１の（①，①）の行を更新しているが，実際には実表Ｒの①の行が更新される。このため，結果的にはビュー表の（①，①）の行のみでなく，（①，②）の行も更新されてしまう。

また，結合キーをそれぞれ別の値で更新した場合にも次のような問題がある。




```
CREATE  VIEW V1 AS
  SELECT * FROM R,S WHERE Rj=Sj
```

V1

	R1	..	Rn	Rj	Sj	S1	..	Sm	
①	a1	..	an	1	1	x1	..	xm	①

```
UPDATE V1 SET Rj=3 Sj=2 WHERE S1=x1
```

V1

	R1	..	Rn	Rj	Sj	S1	..	Sm	
①	a1	..	an	3	3	y1	..	ym	②
②	b1	..	bn	2	2	x1	..	xm	①

上記のような更新を行うと，ビュー表の定義で指定された条件を満たさなくなるだけでなく，それによって他の行と結合される可能性がある。特に，カーソルが位置付けられた時に行が導出されるようなビュー表においては問題となる。

結合結果が１対１となるか，あるいは１対ｎとなるかは，結合する列に対する主キー（以下ＰＫと称する）指定の有無および参照制約定義情報から判定することができる。つまり，ＰＫ同士によるＰＫ－ＰＫ結合であれば結果は１対１であり，参照制約で定義されたＰＫと外部キー（以下ＦＫと称する）によるＰＫ－ＦＫ結合であれば結果は１対ｎとなる。しかし，これを現在のＳＱＬの構文で記述された導出表定義から判定するためには，探索条件として指定されている複数の条件式の中から結合条件を探し出し，そこで指定されている列に対する一意性制約および参照制約定義の有無を調べる必要があり，困難かつ不明瞭である。そこで，ＳＱＬ２で導入された＜結合表＞の定義に，ＰＫ－ＰＫ結合およびＰＫ－ＦＫ結合を明示できるような構文規則を追加した。

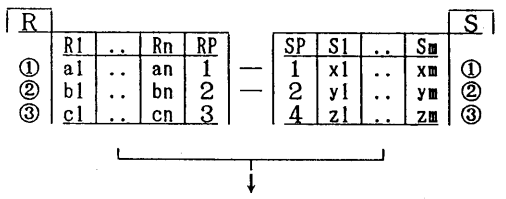
＜結合表＞の結果である導出表を構成する列要素は，結合される表の列を連結したものであるが，今回追加したＰＫ－ＰＫ結合およびＰＫ－ＦＫ結合では，結合の対象となる列（結合キー）は一つに集束され，その名前は先に現れる列の名前が継承される。これは，結合列がそれぞれ別の値で更新されることを防ぐための考慮である。集束された結合キーの値は，導出もとの結合キーのうち，どちらかNULL値でない方の値となる（ＳＱＬ２では COALESCE(RP, SP) と記述する）。

３．２．１　ＰＫ－ＰＫ結合

ＰＫ同士を結合することにより導出される結合表は以下の様に指定する。

表１ [結合ﾀｲﾌﾟ] JOIN 表２ ON PRIMARY KEY

なお，ＰＫ同士の結合表では，集束された結合キーは，結合表自体のＰＫとなる。




```
CREATE V1 AS SELECT * FROM
  (R INNER JOIN S ON PRIMARY KEY)
```

V1

	R1	..	Rn	RP	S1	..	Sm	
①	a1	..	an	1	x1	..	xm	①
②	b1	..	bn	2	y1	..	ym	②

CREATE V2 AS SELECT * FROM
(R LEFT JOIN S ON PRIMARY KEY)

V2

	R1	..	Rn	RP	S1	..	Sm	
①	a1	..	an	1	x1	..	xm	①
②	b1	..	bn	2	y1	..	ym	②
③	c1	..	cn	3	−	..	−	−

CREATE V3 AS SELECT * FROM
(R RIGHT JOIN S ON PRIMARY KEY)

V3

	R1	..	Rn	RP	S1	..	Sm	
①	a1	..	an	1	x1	..	xm	①
②	b1	..	bn	2	y1	..	ym	②
−	−	..	−	4	z1	..	zm	③

CREATE V4 AS SELECT * FROM
(R FULL JOIN S ON PRIMARY KEY)

V4

	R1	..	Rn	RP	S1	..	Sm	
①	a1	..	an	1	x1	..	xm	①
②	b1	..	bn	2	y1	..	ym	②
③	c1	..	cn	3	−	..	−	−
−	−	..	−	4	z1	..	zm	③

ＰＫ－ＰＫ結合表に対する更新規則は，次の通りである。

ＩＮＳＥＲＴ

(1) 指定されたＰＫの値を持つ行が既に導出表に存在してはならない

(2) 指定された値をもつ行がそれぞれの実表に存在しなければ追加する

(1) はＰＫに重複する値を追加できないことを制約している。(2) では片方の表に指定された値をもつ行が既に存在する場合は，その実表に対する追加は必要ないことを示している。

```
INSERT INTO V1 VALUES (i,..,j,1,k,..,l)
=>  例外：不正な更新値

INSERT INTO V1 VALUES (i,..,j,5,k,..,l)
=>  INSERT INTO R VALUES (i,..,j,5)
    INSERT INTO S VALUES (5,k,..,l)

INSERT INTO V2 VALUES (c1,..,cn,3,w1,..,wm)
=>  INSERT INTO V1 VALUES (3,w1,..,wm)

INSERT INTO V2 VALUES (i,..,cn,3,w1,..,wm)
=>  INSERT INTO V2 VALUES (i,...,cn,3)
    =>  例外：一意性制約違反  (i≠c1)
```

ＤＥＬＥＴＥ

(1) 導出もとの行を各表から削除する

結合タイプにＩＮＮＥＲ以外が指定されている＜結合表＞では，行によって導出もとの行が片方の実表にしか存在しない（ＮＵＬＬ拡張されている）場合がある。このような場合，ＮＵＬＬ拡張された側の実表は無視する。

```
DELETE FROM V1 WHERE RP=2
=>  DELETE FROM R WHERE RP=2
    DELETE FROM S WHERE SP=2

DELETE FROM V2 WHERE RP=3
=>  DELETE FROM R WHERE RP=3
```

ＵＰＤＡＴＥ

(1) ＮＵＬＬ拡張された側に更新値が指定された場合指定された値をもつ行を実表に追加する

(2) ＮＵＬＬ拡張されていない側に更新値が指定された場合，導出もとの行を更新する

(3) 結合キーに対して更新値が指定された場合，それぞれの実表に対する更新とする

```
UPDATE V1 SET R1=i S1=j WHERE RP=1
=>  UPDATE R SET R1=i WHERE RP=1
    UPDATE S SET S1=j WHERE SP=1

UPDATE V1 SET RP=9 WHERE RP=1
=>  UPDATE R SET RP=9 WHERE RP=1
    UPDATE S SET SP=9 WHERE SP=1
```

３．２．２　ＰＫ－ＦＫ結合

ＰＫとそれを参照するＦＫとを結合することにより導出される結合表は以下の様に指定する。

```
表１ [結合ﾀｲﾌﾟ] JOIN 表２ ON
 { FOREIGN KEY  |  CONSTRAINT <制約名> }
```

FOREIGN KEY を指定した場合，表１と表２間に唯一の参照制約が定義されていなければならない。CONSTRAINTを指定した場合，＜制約名＞で識別される制約は表１と表２の間に定義された参照制約でなければならない。どちらの場合も，表１と表２の間に定義された参照制約をもとに結合される。ここでは，参照されている側（ＰＫ側）の表を被参照表，参照している側（ＦＫ側）の表を参照表と呼ぶ。
また，ＰＫ－ＦＫ結合では，参照表でＰＫが定義されている場合，それが＜結合表＞の結果として導出される表のＰＫとなる。参照表にＰＫが指定されていなければ，結合表のＰＫは存在しない。

R

	R1	..	Rn	RP
①	a1	..	an	1
②	b1	..	bn	2
③	c1	..	cn	3

S

SF	S1	..	Sm	
1	x1	..	xm	①
1	y1	..	ym	②
−	z1	..	zm	③

CREATE VIEW V1 AS SELECT * FROM
(R INNER JOIN S ON FOREIGN KEY)

V1

	R1	..	Rn	RP	S1	..	Sm	
①	a1	..	an	1	x1	..	xm	①
①	a1	..	an	1	y1	..	ym	②

CREATE VIEW V2 AS SELECT * FROM
(R LEFT JOIN S ON FOREIGN KEY)

V2

	R1	..	Rn	RP	S1	..	Sm	
①	a1	..	an	1	x1	..	xm	①
①	a1	..	an	1	y1	..	ym	②
②	b1	..	bn	2	—	..	—	—
③	c1	..	cn	3	—	..	—	—

CREATE VIEW V3 AS SELECT * FROM
(R RIGHT JOIN S ON FOREIGN KEY)

V3

	R1	..	Rn	RP	S1	..	Sm	
①	a1	..	an	1	x1	..	xm	①
①	a1	..	an	1	y1	..	ym	②
—	—	..	—	—	z1	..	zm	③

CREATE VIEW V4 AS SELECT * FROM
(R FULL JOIN S ON FOREIGN KEY)

V4

	R1	..	Rn	RP	S1	..	Sm	
①	a1	..	an	1	x1	..	xm	①
①	a1	..	an	1	y1	..	ym	②
②	b1	..	bn	2	—	..	—	—
③	c1	..	cn	3	—	..	—	—
—	—	..	—	—	z1	..	zm	③

ＰＫ－ＦＫ結合表に対する更新規則は，次の通りである。

INSERT

(1) 被参照表に指定された値を持つ行が存在しなければ追加する

(2) ＦＫの値がNULL値であり，それ以外の列の値が指定された値と同じである行が参照表に存在する場合，ＦＫの値を指定された値に更新する

(3) そのような行が参照表に存在しない場合，指定された値を持つ行を参照表に追加する

```
INSERT INTO V1 VALUES (i,..,j,5,k,..,l)
=>  INSERT INTO R VALUES (i,..,j,5)
    INSERT INTO S VALUES (5,k,..,l)

INSERT INTO V1 VALUES (b1,..,bn,2,z1,..,zm)
=>  UPDATE S SET SF=2
      WHERE S1=z1 AND ... AND Sm=zm

INSERT INTO V3 VALUES (b1,..,bn,9,z1,..,zm)
=>  INSERT INTO R VALUES (b1,..,bn,9)
    UPDATE S SET SF=9 WHERE F1=z1
```

DELETE

(1) 参照表の行が存在しない場合，披参照表の行を削除する

(2) 被参照表の行と参照表の行が存在し，参照制約定義の参照動作でSET NULLが指定されている場合，ＦＫの値をNULL　値に更新する

(3) それ以外の場合，参照表の行を削除する

参照制約定義では，披参照表の行が削除された場合の動作を指定することが可能であり，その指定によって，参照表の行を削除するか，ＦＫの値をNULL値に更新するかを決定する。なお，参照動作ではSET DEFAULT を指定することも可能であるが，ＦＫを既定値に更新すると他のＰＫと結合される可能性があり，意図しない結果となる場合があるため，ここではSET DEFAULT の指定は無視する。

```
DELETE FROM V1 WHERE RP=1 AND S1=y1
=>  DELETE FROM S WHERE SF=1 AND S1=y1
または
=>  UPDATE S SET SF=NULL
        WHERE SF=1 AND S1=y1

DELETE FROM V2 WHERE RP=2
=>  DELETE FROM R WHERE RP=2
```

UPDATE

(1) 披参照表の列は更新できない

(2) 参照表の行が存在しない場合，指定された値を持つ行を参照表に追加する

(3) それ以外の場合，指定された値で更新する

披参照表の列を更新すると，１対ｎの結合において１側の表を更新することになり，前述したように利用者が意図しない結果となる可能性があるため，このような更新を制限している。また，結合キーは披参照表の列でもあるため，これも更新することはできない。

```
UPDATE V1 SET RP=5
=>  構文エラー

UPDATE V1 SET S1=i, ... , Sm=j
=>  UPDATE S SET S1=i, ... , Sm=j
```

３．３　集合演算

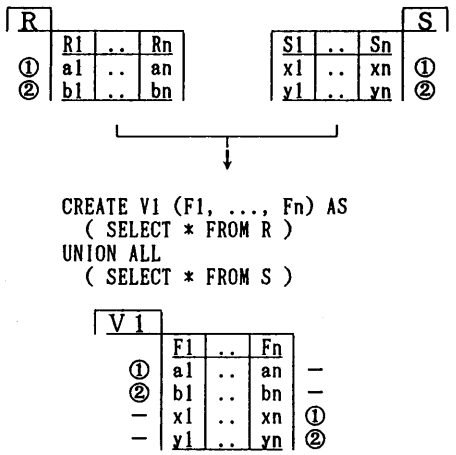
ＳＱＬ１では，カーソル宣言でのみＵＮＩＯＮが指定可能であったが，ＳＱＬ２以降ではビュー定義においてもＵＮＩＯＮが指定可能となっている。また，この他にも，ＩＮＴＥＲＳＥＣＴ，ＥＸＣＥＰＴ，およびＯＵＴＥＲ　ＵＮＩＯＮといった集合演算が追加されている。

これらの集合演算では，ＡＬＬの指定がオプションとなっており，これが指定されていない場合は集合演算結果から重複する値を持つ行は１行しか含まれない。このため，ＡＬＬ指定のない集合演算については，更新の対象となる実表の行が一意に決定できない場合があり，このような集合演算の結果として導出される表については更新不可とする。

ＡＬＬ指定のある集合演算結果として導出される導出表については，結合結果と同様にＴＩＤの組で保持することができ，対象となる実表の行を一意に決定できる。ただし，集合演算によっては，実表に対する更新が何通りも存在する場合があり，そのような更新は制限しなければならない。

３．３．１　ＵＮＩＯＮ

UNION ALL の結果となる導出表には，集合和として導出もとの表の各行が含まれる。



ＩＮＳＥＲＴ

(1) 導出表に行を追加することはできない

集合和の導出表に対する行の追加は，どちらの表に対する追加を意図しているのかが不明である。

ＤＥＬＥＴＥ

(1) 導出もとの表から行を削除する

```
DELETE FROM V1 WHERE F1=a1
=>  DELETE FROM R WHERE R1=a1

DELETE FROM V1 WHERE F1=x1
=>  DELETE FROM S WHERE S1=x1
```

ＵＰＤＡＴＥ

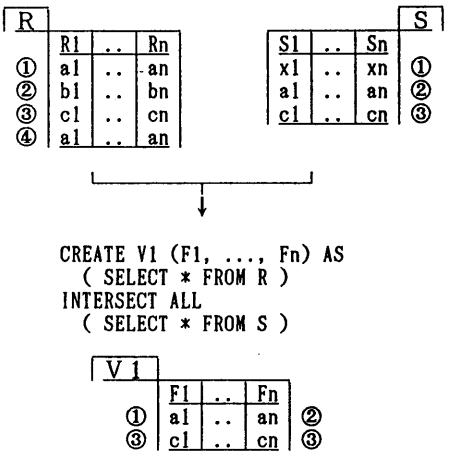
(1) 導出もとの表の行を更新する

```
UPDATE V1 SET F1=w1 WHERE F1=a1
=>  UPDATE R SET R1=w1 WHERE R1=a1

UPDATE V1 SET F1=w1 WHERE F1=x1
=>  UPDATE S SET S1=w1 WHERE S1=x1
```

３．３．２　ＩＮＴＥＲＳＥＣＴ

INTERSECT ALL の結果となる導出表には，集合積として導出もとの実表の両方に存在する行が含まれる。



ＩＮＳＥＲＴ

(1) 指定された値を持ち，既に導出表に含まれていない行が実表に存在しなければ，実表に追加する。

```
INSERT INTO V1 VALUES (i, ..., k)
=>  INSERT INTO R VALUES (i, ..., k)
    INSERT INTO S VALUES (i, ..., k)

INSERT INTO V1 VALUES (a1, ..., an)
=>  INSERT INTO S VALUES (a1, ..., an)
```

ＤＥＬＥＴＥ

(1) 導出表から行を削除することはできない

集合積の導出表に対する行の削除は，どちらの表に対する削除を意図しているのかが不明である。

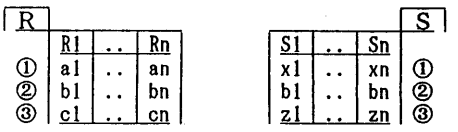
ＵＰＤＡＴＥ

(1) 対応する実表の行を両方とも更新する

```
UPDATE V1 SET F1=w1 WHERE F1=a1
=>  UPDATE R SET R1=w1 WHERE R1=a1
    UPDATE S SET S1=w1 WHERE S1=a1
```

３．３．３　ＥＸＣＥＰＴ

EXCEPT ALLの結果となる導出表には，集合差として１番目に指定された実表の行で，２番目に指定された実表に存在しない行が含まれる。



```
CREATE V1 (F1, ..., Fn) AS
  ( SELECT * FROM R )
EXCEPT ALL
  ( SELECT * FROM S )
```

V1

	F1	..	Fn
①	a1	..	an
③	c1	..	cn

INSERT

(1) 導出表に行を追加することはできない

集合差の導出表に対する行の追加は，１番目の表に対する追加を意図しているのか，２番目の表からの削除を意図しているのかが不明である。

DELETE

(1) 導出表から行を削除することはできない

集合差の導出表に対する行の削除は，１番目の表からの削除を意図しているのか，２番目の表に対する追加を意図しているのかが不明である。

UPDATE

(1) 導出もとの実表の行を更新する

```
UPDATE V1 SET F1=w1 WHERE F1=a1
=>  UPDATE R SET R1=w1 WHERE R1=a1
```

3．3．4 OUTER UNION

OUTER UNION ALL の結果は導出表には，導出もとの実表の各行が，それぞれＮＵＬＬ拡張された形で含まれる。

R

	R1	..	Rn
①	a1	..	an
②	b1	..	bn

S

S1	..	Sm	
x1	..	xm	①
y1	..	ym	②

```
CREATE V1 (F1, ..., Fn+m) AS
  ( SELECT * FROM R )
OUTER UNION ALL
  ( SELECT * FROM S )
```

V1

	F1	..	Fn	Fn+1	..	Fn+m	
①	a1	..	an	—	..	—	—
②	b1	..	bn	—	..	—	—
—	—	..	—	x1	..	xm	①
—	—	..	—	y1	..	ym	②

INSERT

(1) １番目の表から導出された列と２番目の表から導出された列の両方にNULL値以外の値を指定してはならない

(2) １番目の表から導出された列に対して指定された値が全てNULL値の場合，指定された値を持つ行を２番目の表に追加する

(3) それ以外の場合，指定された値を持つ行を１番目の表に追加する

OUTER UNION から成る導出表に含まれる行はすべてNULL拡張された行であるため，１番目の表から導出された列と２番目の表から導出された列の両方にNULL値以外の値を指定することはできない。

```
INSERT INTO V1 VALUES (i,..,j,k,..,l)
=>  構文エラー  (i≠NULL AND l≠NULL)

INSERT INTO V1 VALUES (NULL,..,NULL,k,..,l)
=>  INSERT INTO S VALUES (k,..,l)

INSERT INTO V1 VALUES (i,..,j,NULL,..,NULL)
=>  INSERT INTO R VALUES (i,..,j)
```

DELETE

(1) 導出もとの実表から行を削除する

```
DELETE FROM V1 WHERE F1=a1
=>  DELETE FROM R WHERE R1=a1

DELETE FROM V1 WHERE Fn+1=x1
=>  DELETE FROM S WHERE R1=x1
```

UPDATE

(1) ＮＵＬＬ拡張された列に対して更新値は指定できない

(2) 導出もとの実表の行を指定された値で更新する

ＮＵＬＬ拡張された側の実表に対する更新は不可とするが，行によってＮＵＬＬ拡張されている側が異なるため，これは実行時に例外とする。

```
UPDATE V1 SET Fn+1=w1 WHERE F1=a1
=>  例外：不正更新値

UPDATE V1 SET F1=w1 WHERE F1=a1
=>  UPDATE R SET R1=w1 WHERE R1=a1
```

3．4 ORDER BY

ＯＲＤＥＲ　ＢＹ句の指定によって順序付けられた＜カーソル宣言＞では，実表のＴＩＤを導出表中に出現する順番で保持することにより，導出表の行と実表の行とを１対１に対応付けることが可能である。

R

	R1	..	Rn
①	3	..	an
②	1	..	bn
③	2	..	cn

```
DECLARE X CURSOR FOR
  SELECT * FROM R
ORDER BY R1
```

X

	R1	..	Rn
③	1	..	cn
①	2	..	an
②	3	..	bn

ここで問題になるのは，ソートキーに対する更新であり，このような更新を許した場合の影響が不明確である。例えば，ＯＲＤＥＲ　ＢＹ句で指定された列に対する索引が存在し，その索引を利用して行を取り出す場合，その列の値を更新すると，後続の行を取り出す過程でその行が再び出現する可能性がある。これに対し，索引を使用せず，ソートした順番でＴＩＤを保持する場合，同じ行が再度出現するようなことはない。

このように，実装方式によって結果が異なる危険性を避けるため，ＯＲＤＥＲ　ＢＹ句指定のあるカーソルについては，ソートキー以外の列に限り更新可能とする。

4．むすび

本稿で述べてきた更新規則は，カーソル／ビューに対する更新範囲をどこまで拡張できるかを検討し，データベース言語ＳＱＬの構文規則として記述したものである。本提案は，ＩＳＯのＪＴＣ１／ＳＣ２１／ＷＧ３　ＤＢＬ委員会において，ＳＱＬ３機能として受け入れられており，現在の作業ドキュメントにも含まれている。

ただし，これでカーソル／ビューの更新問題がすべて解決された訳ではなく，更新可能でないカーソル／ビューも依然として数多く残っている。また，ここで述べきた更新規則に対して，異なる見解も当然存在するであろう。そういった意味では，本提案のカーソル／ビュー更新規則はまだ完全であるとは言えず，改良の余地が残されている。しかし，これまでビュー更新問題が，どちらかと言うと，概念的な側面からの議論に終始してきたことを考えた場合，標準データベース言語であるＳＱＬの構文規則としてビュー更新規則が記述されたことによって，この問題をより現実的なものとしてとらえ，検討していくための基盤を確立することができたという点で，非常に大きな意味があると考える。

今後もカーソル／ビュー更新規則についての検討を継続し，より完全なものに改良していくとともに，利用者の意図を明確に示す構文等の導入により，さらに更新可能なカーソル／ビューの範囲を拡張する方法について検討していきたい。

参考文献

[1] 日本規格協会: 日本工業規格データベース言語ＳＱＬ，JIS X3055-1987.

[2] ISO SC21 N3155: Draft Proposed Database Language SQL2, Jan. 1989.

[3] ISO-ANSI (working draft) Database Language SQL2 and SQL3, Feb. 1989.

[4] C. J. Date: Relational Database Selected Writings, Addison-Wesley 1986.

[5] 増永，野口: 関係データベースビュー更新問題の意味論的解決法，情報処理学会 Vol.25 No.1 Jan. 1984.