日本データベース学会和文論文誌
Vol.19-J, Article No.1, 2021年3月

整合ラベリング問題としてのクロス結合ビューの更新可能性

増永 良文♥　長田 悠吾♠　石井 達夫♦

結合ビューはリレーショナルデータベースのアプリケーション開発にとって大変有用であるが，射影ビューや選択ビューと異なり，その更新は理論上も実用上もほとんどの場合不可とされている．増永ら[7]はその制約がこれまでビュー更新問題をデータベースのスキーマレベルで論じてきたことに起因し，そうではなくそれをデータベースのインスタンスレベルで論じると，従来のアプローチでは更新不可とされてきた結合ビューが場合により更新可能となることを示している．この新しい手法は「意図に基づくアプローチ」と名付けられている．しかしながら，そのアプローチはそれが集合意味論の下で形式化されていること，そして結合ビューの更新可能性を力まかせ法で検証していることに改善の余地があった．そこで本論文では，まずこのアプローチをバッグ意味論の下で形式化し直した場合のビューサポート能力を明らかにし，続いてこの一般的な状況下において，クロス結合ビューへの更新操作の変換可能性が制約充足問題の一つである整合ラベリング問題に帰着でき，その解法の一つである併合法を用いて解けることを示す．この手法を CLP 法と名付けるが，クロス結合ビューの更新可能性を力まかせ法と CLP 法で検証する場合の計算量について考察した結果を示す．

1. はじめに

結合ビュー（join view）はリレーショナルデータベースのアプリケーション開発において選択ビューや射影ビューと共に大変有用である．しかしながら，ビューは仮想的なリレーションであるために，一般にビューを更新しようとすると強い制約が掛かり，従来の理論では結合ビューはほとんどの場合で更新不可とされ，また実用面でも SQL:1999[5]で初めて更新可能な結合ビューが規格化されたものの，Oracle Database での実装[9]に見るように更新可能な結合ビューの条件は極めて限定的である．増永ら[7]はその原因を究明し，これまでの研究がこの問題をデータベースのスキーマレベルで論じてきたことにあり，そうではなくそれをデータベースのインスタンスレベルで論じると結合ビューが場合により更新可能となることを示している．この新規アプローチは「更新意図の外形的推測に基づくアプローチ」，略して「意図に基づくアプローチ」と名付けられ，その実現可能性を PostgreSQL 上で検証している．しかしながら，このアプローチは集合意味論(set semantics)に基づいて形式化されている点や結合ビューの更新可能性検証を力まかせ法(brute force)を用いて検証している点などに改善の余地があった．

そこで，本論文では，これらの問題点を解決するべく，基礎的事項を与えた後(第 2 章)，意図に基づくアプローチが国際標準リレーショナルデータベース言語 SQL(以下，SQL)が依って立つバッグ意味論(bag semantics)の下でどのようなビューサポート能力を持つのかを検証してその結果を報告すると共に，HISU(Hybrid Instance-level/Schema-level Updatability, 混合インスタンスレベル／スキーマレベル更新可能性)と名付けられたビュー更新法を提案する(第 3 章)．続いて，論点をクロス結合ビュー(＝直積ビュー)の更新可能性の検証問題に特化し，この問題が非線形連立方程式を解く問題に帰着されること，更にこの問題が制約充足問題(constraint satisfaction problem, CSP)の一つである整合ラベリング問題(consistent labelling problem, CLP)に帰着できること，そしてこの問題はその解法の一つとして知られている併合法(merge method)を用いることで解くことができることを示す(第 4 章)．このアプローチを CLP 法と名付けるが，局所解を SQL の 2 項テーブル表現できることに着目すると，CLP 法の計算量はリレーショナル DBMS 依存であることが示される(第 5 章)．第 6 章はまとめである．

2. 基礎的事項

2.1 バッグ

リレーショナルデータモデルはリレーションやビュー，そしてデータ操作言語を集合意味論の下で形式化しているが，SQL はテーブルやビューに重複する行の出現を許すバッグ意味論の下で言語設計がなされている．そこで，バッグ（bag），マルチ集合(multiset)[1]ともいう，を定義することから始める．

【定義 1】(バッグ) $\boldsymbol{R}(A_1, A_2, ..., A_n)$をリレーションスキーマとする．$\Omega_{\boldsymbol{R}}=\{A_1, A_2, ..., A_n\}$を$\boldsymbol{R}$の全属性集合，dom をドメイン関数とし，$dom(\boldsymbol{R})=dom(A_1)\times dom(A_2)\times\cdots\times dom(A_n)$とする．このとき，$\boldsymbol{R}$のバッグ R を次のように定義する．

$$R(A_1, A_2, ..., A_n)=\{t_1(k_1), t_2(k_2), ..., t_p(k_p)\}$$ [1]

ここに$(\forall i=1,..,p)(t_i\in dom(\boldsymbol{R})\wedge k_i\geqq 1)$とする．$k_i$は R 中に生起する行(row)$t_i$の重複度（multiplicity）を表す．

R の濃度（cardinality）を$|R|=\Sigma_{i=1,...,p}k_i$と定義する．バッグ R は$(\forall i=1,..,p)(k_i=1)$のときリレーション(＝集合)である．なお，$k_i=1$のとき，$t_i(1)$と書かずに，単に$t_i$と書くことがある．また$t_i(0)$は$t_i$が存在していないことを意味する．

2.2 バッグ代数

一般に，$R(A_1, A_2, ..., A_n)=\{t_1(k_1), t_2(k_2), ..., t_p(k_p)\}$と$S(B_1, B_2, ..., B_m)=\{u_1(\ell_1), u_2(\ell_2), ..., u_q(\ell_q)\}$をバッグとするとき，バッグ代数は次の 8 個の演算子からなる[3]：重複行除去（δ），和[2]($\uplus$)，差(−)，共通($\cap$)，クロス結合(＝直積)(×)，射影(R[X]，ここに$X\subseteq\Omega_{\boldsymbol{R}}$)，$\theta$-選択($R[A_i\,\theta\,A_j]$，ここに，R の属性$A_i$と$A_j$は$\theta$-比較可能)，$\theta$-結合($R[A_i\,\theta\,B_j]S$，ここに，$R.A_i$と$S.B_j$は$\theta$-比較可能)．

補足すれば，$\delta(R)=\{t_1, t_2, ..., t_p\}$であり，和，差，共通演算では R と S は和両立[3]と仮定する．なお，和，差，共通演算はそれぞれ SQL の UNION ALL, EXCEPT ALL, INTERSECT

♥ 正会員　お茶の水女子大学名誉教授 masunaga.yoshifumi@ocha.ac.jp
♠ 正会員　SRA OSS, Inc.日本支社 nagata@sraoss.co.jp
♦ 正会員　SRA OSS, Inc.日本支社 ishii@sraoss.co.jp

[1] $t_i(k_i)$はバッグ R に行t_iが丁度k_i本あることを表すための記法で，この記法を用いて表現されたバッグを「重複度付き行表現標準形」ということにする．このとき，$t_i(k_i)\in R$と書く．本論文では，原則，バッグはこの標準形で与えられる．

[2] マルチ集合論では最大和($\cup$)と加法和($\uplus$)の 2 種類の和演算が定義されている．違いを端的に表すと，$\{1\}\cup\{1(2)\}=\{1(2)\}$であり，$\{1\}\uplus\{1(2)\}=\{1(3)\}$である．最大和は集合意味論での和と整合する．加法和はそうでないが SQL と整合する．

[3] 一般に，バッグ$R(A_1, A_2, ..., A_n)=\{t_1(k_1), t_2(k_2), ..., t_p(k_p)\}$とバッグ$S(B_1, B_2, ..., B_m)=\{u_1(\ell_1), u_2(\ell_2), ..., u_q(\ell_q)\}$が和両立とは，$n=m\wedge(\forall i=1, ..., n)(dom(A_i)=dom(B_i))$が成立するときをいうが，R と S が共に$\boldsymbol{R}$(あるいは$\boldsymbol{S}$)のバッグであればこの条件は満たされるので，以下，このように仮定する．

ALLに対応し，クロス結合，θ-選択，θ-結合の演算結果からは重複行は削除されず SELECT ALL の意味を持つ．上記8個の演算はお互いに独立ではなく，例えば $R\cap S=R-(R-S)$であり，$R[A_i\,\theta\,B_j]S=(R\times S)[R.A_i\,\theta\,S.B_j]$である．リレーショナル代数演算の独立性と同様に，バッグ代数の重複行除去，和，差，クロス結合，射影，選択の6つの演算は互いに独立である．独立でない演算子を許容しているのはその有用性による．

演算は再帰的に適用可能であり，リレーショナル代数表現と同様に，問合せはバッグ代数表現(bag algebra expression)として書き表せる．

2.3 バッグの更新操作

更新(update)とはバッグから一般に複数の行の削除(delete)，挿入(insert)，あるいは書換(rewrite)を意味する．バッグRに対する削除と挿入操作は次のように定義される[3]：

削除操作 d: **delete** D **from** R;

ここに，Dはリレーションスキーマ $\boldsymbol{R}$のバッグとし，削除結果はR−Dである．

挿入操作 i: **insert into** R **values** I;

ここに，Iは $\boldsymbol{R}$のバッグとし，挿入結果は $R\uplus I$ である[4]．

なお，書換操作 r: **rewrite** D **of** R **to** I;，ここにDとIは $\boldsymbol{R}$のバッグとする，はDの削除操作とIの挿入操作の系列と定義する．

2.4 ビュー

ビューは次のように定義される．

【定義2】(ビュー)

1. 実バッグ[5]Rはビューである．
2. Vをビューとするとき，δ(V)はビューである．このビューを重複行削除ビューという．
3. V_1，V_2をビューとするとき，V_1とV_2が和両立ならば，$V_1\uplus V_2$，V_1-V_2，$V_1\cap V_2$もビューである．これらのビューをそれぞれ和ビュー，差ビュー，共通ビューという．
4. V_1，V_2をビューとするとき，$V_1\times V_2$はビューである．これをクロス結合ビューという．
5. Vをビューとするとき，V[X]はビューである．ここに，XはVの属性集合である．これを射影ビューという．
6. Vをビューとするとき，$V[A_i\,\theta\,A_j]$はビューである．ここに，A_iとA_jはθ-比較可能とする．これをθ-選択ビューという．
7. V_1，V_2をビューとするとき，$V_1[A_i\,\theta\,B_j]V_2$はビューである．ここに，$V_1.A_i$と$V_2.B_j$はθ-比較可能とする．これをθ-結合ビューという．
8. 1. から7. で定義されるもののみが，ビューである．

このように，一般にビューはバッグ代数表現として再帰的に定義されるが，前提となるビュー(VやV_1，V_2)が実バッグであるとき，定義されたビューを「基本ビュー」ということにする．なお，自然結合ビュー$V_1{*}V_2$など陽には定義されていないビューも上述のビュー定義演算を再帰的に適用することで導ける．

2.5 ビューへの更新操作の変換可能性

ビューへの更新操作の変換可能性は次のように定義される[2]：

【定義3】(ビューへの更新操作の変換可能性)

ビューはデータベース状態からビュー状態への関数である[6]．s_τをある時刻 τ におけるデータベース状態，Vをビュー定義，$V(s_\tau)$は(その時刻 τ における)ビュー状態，$\boldsymbol{u}$を$V(s_\tau)$に対して発行された更新操作とする．このとき，$\boldsymbol{u}$が変換可能(translatable)であるとは，$\boldsymbol{u}$をs_τへの更新操作に変換する(i) 一意(unique)で，(ii) 副作用がない(no side effects)変換Tが存在するときをいう．条件(i)で変換Tに一意性を課した理由は，Tに代替案があった時，その選択基準を設けられないためである．(ii)の変換Tに副作用がないとは，$\boldsymbol{u}(V(s_\tau))=V(T(\boldsymbol{u})(s_\tau))$ が成立することをいう．すなわち，$\boldsymbol{u}$が変換可能であるとは，図1の可換図式が成立する一意な変換Tが存在するときをいう．

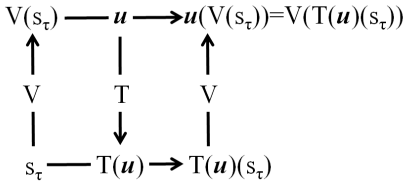

図1　ビューへの更新操作 u が変換可能であるための可換図式

なお，(i)と(ii)に加えて，(iii) 変換がデータベースに対して無関係な(extraneous)更新を掛けないという条件を課すことができるが[7]，本論文ではビューに対する更新操作の変換は，所望のビュー更新を実現させるために必要な操作のみを変換プロセスの対象とするので，無関係な更新が掛かることはなく，この条件は課さなくてよい．

ビューへの更新操作の変換可能性を「ビューの更新可能性」ということも多く，本論文でもその慣例に従う．

2.6 スキーマレベル vs. インスタンスレベルアプローチ

ビューの更新可能性に関する研究はこれまで数多く報告されているが，大別すると構文的アプローチ[2]，意味的アプローチ[6]，対話的アプローチ[10]に分類できる．しかし，これらの研究に共通する特徴としてビューの更新可能性をデータベースの“スキーマレベル”で形式化している点を指摘できる．SQLでのビューサポートもその例外ではない．

一方，「意図に基づくアプローチ」はビューの更新可能性をデータベースの“インスタンスレベル”で形式化している．その結果，スキーマレベルでは更新不可とされた結合ビューが場合によって，つまりビュー状態と発行された更新操作の組合せによって，更新可能となり，更新可能性が向上することが報告されている[7]．両アプローチの違いを例題を交えて以下に示す．

2.6.1 スキーマレベルアプローチによる更新可能性

ビューの更新可能性をスキーマレベルで規定するということは，例えば，“自然結合ビューは削除可能か？”という命題が建てられ，

---

[4] 結果は，重複度付き行表現標準形に整形したものとする．以下同様．

[5] 実バッグとはデータベースに格納されているバッグのことをいう．

[6] データベース状態(state)とはデータベースを構成している実テーブル群のその時刻でのインスタンス（＝スナップショット)のことをいい，ビュー状態とはそれらのインスタンスを基にマテリアライズ(materialize)されたビューのインスタンスのことをいう．

[7] 例えば，社員テーブル 社員(社員名，給与，年齢)の{社員名，給与}上の射影ビュー 社員給与(社員名，給与)に対して，挿入操作 $\boldsymbol{i}$=insert into 社員給与 values (山田太郎, 50)が発行されたとき，年齢を空(—, null)で補完して，$\boldsymbol{i}$を$T_1(\boldsymbol{i})$=insert into 社員 values (山田太郎, 50, —);と変換することで$\boldsymbol{i}$は変換可能となるが（これが本論文の変換法），年齢値を適当に補完して，例えば，$T_2(\boldsymbol{i})$=insert into 社員 values (山田太郎, 50, 40);などとはしないということ．このように変換しても可換図式は成立する．

それに対して 1 つでも削除可能でない自然結合ビューのインスタンスとそれに対して発行される削除操作の存在が示されれば，“自然結合ビューは削除可能ではない”と結論付けるということである．このことをクロス結合ビューの更新可能性で例示してみると次のようである．

【例題 1】バッグ R(A)と S(B)のクロス結合ビューを V=R(A)×S(B)とし，V に対して次の削除操作が発行されたとする．

d: **delete** D **from** V;

この操作が発行された時点での R と S のインスタンスは R(A)={1(2)}, S(B)={1(2)}で，D={(1, 1)(2)}とする．すると，容易に確認できるように，***d***は次に示す何れの削除操作でも実現できる．

T_1(***d***): **delete** 1 **from** R;

T_2(***d***): **delete** 1 **from** S;

しかしながら，この変換の曖昧性を解消することはできないから，ビューの更新可能性の条件(i)に抵触し，更新不可である．そして，このように削除不可となるクロス結合ビューのインスタンスとそれに対して発行された削除操作の組が見つかったので“クロス結合ビューは削除不可”と結論される．クロス結合ビューに対する挿入操作も同様に変換不可と結論され，従って更新不可である．

2.6.2 インスタンスレベルアプローチによる更新可能性

例題 1 で示したように，確かに変換の曖昧性を解消できない場合もあるが，場合によっては，複数の変換候補を一つひとつインスタンスレベルで検証していくと，ただ一つの候補だけが図 1 の“可換性”を満たせることがあり，それであればそれをユーザの更新意図であろうと“推測”して，その変換でビューを更新しようという発想が湧く．それがインスタンスレベルアプローチ，すなわち「意図に基づくアプローチ」である．なお，意図に基づくアプローチの妥当性を 3.7 節で論じている．以下に，意図に基づくアプローチがビューの更新可能性をどのように検証するかを例題で示す．ここでは変換の唯一性を検証するために「力まかせ法」を用いているが，本論文 4 章ではそれに代わる「CLP 法」を提示し，5 章では両者の計算量について考察している．

[1] クロス結合ビューの削除可能性

一般に R(A)={$t_1(k_1)$, $t_2(k_2)$, ..., $t_p(k_p)$}と S(B)={$u_1(\ell_1)$, $u_2(\ell_2)$, ..., $u_q(\ell_q)$}，ここに A と B はリレーションスキーマ ***R*** と ***S*** の全属性集合を表すとする，をバッグとし，V=R×S をそれらのクロス結合ビューとする．このとき，V に対して削除操作 ***d***: **delete** D **from** V; が発行されたとする．ここに，D はリレーションスキーマ ***V***(=***R***×***S***)のバッグである[8]．このとき，もし ***d***が変換可能であれば一般に R のバッグ R^d，S のバッグ S^dが存在して，(V−D)=(R−R^d)×(S−S^d)となるはずである[9]．従って，***d*** が変換可能かどうかは，そのような R^dと S^dが一意に同定できるか否かという問題に帰着される．従って，次を示せる．

■ クロス結合ビューの削除可能性検証法

1. V をマテリアライズする．
2. D={$u_1(k_1)$, $u_2(k_2)$, ..., $u_p(k_p)$}とする．
3. R^dと S^dのあらゆる候補を列挙し，(V−D)=(R−R^d)×(S−S^d)が成立するか否かを検証する．等号が成立する R^dと S^dの組がただ一つ存在すれば ***d***は変換可能である．

なお，V と D から R^dと S^dのあらゆる候補を列挙する手法は次の通りである．

- δ(D[A])∩δ(R)={$t_{1'}$, $t_{2'}$, ..., $t_{q'}$}とする．$t_{i'}(k_{i'})$∈R とすれば，***d***が変換可能であるためには R から $t_{i'}$を 0〜$k_{i'}$本削除する必要がある．それが各 $t_{i'}$について言えるから，可能なR^dの候補数はそれらの組合せで$\Pi_{i=1, ..., q}(k_{i'}+1)$となる．$S^d$についても同様である．それらの数をそれぞれ $C^{R\text{-}d}$, $C^{S\text{-}d}$とすれば，ステップ 3.での検証数は $C^{R\text{-}d}\times C^{S\text{-}d}$となる．

この検証法の正当性はその構成から明らかであろう．例題で補う．

【例題 2】R(A)={1(1), 2(2)}, S(B)={10(1), 20(3)}を実バッグ，R と S のクロス結合ビューを V=R(A)×S(B)とする．V をマテリアライズすると{(1,10)(1), (1,20)(3), (2,10)(2), (2,20)(6)}である．V に対して次の削除操作 ***d***: **delete** D **from** V; ここに，D={(2, 10)(2), (2, 20)(6)}とする，が発行されたとする．すると，δ(D[A])={2}, δ(D[B])={10, 20}である．従って，***d*** が変換可能であるとするならば，R(A)からは行 2 を 0〜2 本削除する 3 通り，S(B)からは行 10 を 0〜1 本削除する 2 通りと，行 20 を 0〜3 本削除する 4 通りとの組合せである 8 通り，従って 24(=3×8)通りの組合せに対して，(V−D)=(R−R^d)×(S−S^d)が成立するかどうかを検証すればよい．この場合，R^d={2(2)}, S^d=∅(空)の組合せで唯一等号が成立するので，***d*** を **delete** {2(2)} **from** R; に変換すればよいことが分かる．

[2] クロス結合ビューの挿入可能性

R と S のクロス結合ビューV に対して挿入操作 ***i***: **insert into** V **values** I; が発行されたとする．ここに，I はリレーションスキーマ ***V***(=***R***×***S***)のバッグである．この挿入操作の結果は V⊎I であるが，***i***が変換可能であるためには，***R***のバッグ R^i，***S***のバッグ S^iが存在して，(V⊎I)=(R⊎R^i)×(S⊎S^i)とならねばならない．従って，***i***が変換可能かどうかは，そのような R^iと S^iが一意に同定できるか否かという問題となる．

■ クロス結合ビューの挿入可能性検証法

1. V をマテリアライズする．
2. I={$v_1(k_1)$, $v_2(k_2)$, ..., $v_p(k_p)$}とする．
3. R^iと S^iのあらゆる候補を列挙し，(V⊎I)=(R⊎R^i)×(S⊎S^i)が成立するか否かを検証する．等号が成立する R^iと S^iの組がただ一つ存在すれば ***i***は変換可能である．

なお，V と I から R^iと S^iのあらゆる候補を列挙する手法は次の通りである．

- I[A]と I[B]を作成する．まず，δ(I[A])={$w_{1'}$, $w_{2'}$, ..., $w_{r'}$} とし，$w_{i'}(k_{i'})$∈I[A]とする．このとき，I が変換可能となるためには，R に $w_{i'}$を 0〜$k_{i'}$本，新規(δ(I[A])∩δ(R)=∅のとき)あるいは追加(≠∅のとき)で挿入しないといけない．それが各 $w_{i'}$について言えるから，可能な R^iの候補数はそれらの組合せで$\Pi_{i=1, ..., q}(k_{i'}+1)$となる．I[B]についても同様である．そこで，$R^i$と S^iの候補数をそれぞれ $C^{R\text{-}i}$, $C^{S\text{-}i}$とすれば，ステップ 3.での検証数は $C^{R\text{-}i}\times C^{S\text{-}i}$となる．

この検証法の正当性もその構成から明らかであろう．例で補う．

【例題 3】R(A), S(B), それらのクロス結合ビューを V, は例題 2 で与えた通りとする．V に対して次の挿入操作が発行されたとする：***i***: **insert into** V **values** I; ここに，I={(1, 30)(2), (2, 30)(4)}とする．このとき，I[A]={1(2), 2(4)}, I[B]={30(6)}である．***i***が変換可能であるとするならば，R(A)に行 1 を 0〜2 本追加挿入す

[8] R(A_1, A_2, ..., A_n), S(B_1, B_2, ..., B_m)をリレーションスキーマとするとき，V=R×S のスキーマは ***V***(A_1, A_2, ..., A_n, B_1, B_2, ..., B_m)である．

[9] V−D は V をマテリアライズして D との差を計算することで得られる．

る3通り，行2を0～4本追加挿入する5通り，計15(=3×5)通りの候補があり，S(B)については行30を0～6本新規挿入する7通り可能性が存在するから，従って，計105(=15×7)通りの候補がある．それぞれについて(V⊎I)=(R⊎R^i)×(S⊎S^i)が成立するか否かを検証すると，R^i=∅とS^i={30(2)}の組でのみ等号が成立することが分かるから，***i***は変換可能であり，それは **insert into** S **values** {30(2)};に変換すればよいことが分かる．

以上，クロス結合ビューは場合により削除可能，並びに挿入可能であることが示された．従って，場合により書換可能となる，なお，書換の場合は書換操作の両立という条件が課される[10]．詳細は[7]を参照されたい．

3. バッグ意味論の下でのビューの更新可能性

バッグ意味論の下で定義された8つの基本ビューの更新可能性を論じる．この際，脚注2で述べたが，集合意味論での和はバッグ意味論での和の特殊な場合とはならないことに注意する．それ以外の演算はそれらの定義からこの意味での問題はない．そして，明確にしておかねばならないことは，ビューの更新可能性がスキーマレベルアプローチの結果であるのか，インスタンスレベルアプローチの結果であるのかである．なお，8つの基本演算を使って再帰的に定義される一般的なビューの更新可能性をどのように判定するかは，θ-結合ビューの更新可能性を論じた3.6節で説明する．なお，ビューが削除不可か挿入不可であれば書換不可である．

3.1 重複行除去ビューの更新可能性

重複行除去演算 δ はバッグ代数に特有な演算である．一般にバッグ $R=\{t_1(k_1), t_2(k_2), ..., t_p(k_p)\}$の重複行除去ビュー$\delta(R)=\{t_1, t_2, ..., t_p\}$に対して，(i) $\delta(R)$への t_i の削除操作はRへの $t_i(k_i)$の削除操作に一意に副作用なく変換可能である．(ii) $\delta(R)$へのタップル t の挿入は，R に t を一本以上何本挿入してもそれを実現できるから，変換は一意ではなく変換不可である．(i)と(ii)はすべてのバッグRに対して成り立つから，スキーマレベルの性質である．

3.2 バッグ意味論の下での和，差，共通ビューの更新可能性

集合はバッグの特殊な場合であるから集合意味論の下で変換不可とされた更新操作がバッグ意味論の下で変換可能となることはない(逆はありうる)．従って，和，差，共通ビューについては，集合意味論の下では変換可能であった和―削除，差―挿入，共通―挿入の組合せ[7]について，それらがバッグ意味論の下でも変換可能か否かが検証の対象となる．

3.2.1 和―削除の場合

RとSを和両立なバッグとし(一般性を失うことなく，共にリレーションスキーマ ***R*** のバッグとする)，V=R⊎SをRとSの和ビューとする．その定義から次が成り立つ．

∀R∀S∀t∀k(t(k)∈V iff ∃m∃n(t(m)∈R∧t(n)∈S∧k=m+n))

このとき，Dを***R***のバッグとし，削除操作 ***d***: **delete** D **from** V;が発行されたとする．削除の定義から，***d***が発行された時点での和ビューのインスタンスをVとすれば，t(k)∈V∩D，ここにk=m+nでm≠0∧n≠0，をVから削除しようとするとき，和ビューVの定義によりRからt(m)を削除する，Sからt(n)を削除する，k=m+nが成り立たないようにRやSを更新する，及びこれらの組合せで実現できることになる．このように変換の曖昧性が生じるから変換不可である．これはすべてのR, S, Dに対して成り立つからスキーマレベルの性質である．

なお，m≠0∧n≠0の条件を外すと，m=0∨n=0が成立することになる．例えば，m=0であれば，k=nということになるので，この場合にはt(k)∈V∩DをVから削除する操作は一意に副作用なくSからt(n)を削除するという操作に変換可能となる．n=0の場合も同様である．これもスキーマレベルの性質である．

3.2.2 差―挿入の場合

RとSを和両立なバッグとし，V=R−SをRとSの差ビューとする．その定義から次が成り立つ．

∀R∀S∀t∀k(t(k)∈V iff ∃m∃n(t(m)∈R∧t(n)∈S∧k=max(0, m−n))，ここにmax(0, m−n)=m−n (m≧n)，=0 (その他)

このとき，Iを***R***のバッグとし，挿入操作 ***i***: **insert into** V **values** I;が発行されたとすると，挿入の定義から，***i***が発行された時点でのVのインスタンスをVとすれば，***i***(V)=V⊎Iである．このとき，$\delta(V)\cap\delta(I)=\emptyset$か$\neq\emptyset$の2つの場合が想定されるが，何れの場合もt(k)∈IをVに挿入するにあたりk=m−nのmとnをどのように振り分けるかに任意性があり，***i***の変換に曖昧性が生じて変換不可である．この性質はすべてのR, S, Iについて成り立つからスキーマレベルである．

3.2.3 共通―挿入の場合

RとSを和両立なバッグとし，V=R∩SをRとSの共通ビューとする．その定義から次が成り立つ．

∀R∀S∀t∀k(t(k)∈V iff ∃m∃n(t(m)∈R∧t(n)∈S∧k=min(m, n))，ここにmin(m, n)=m (m≦n)，=n (その他)

このとき，Iを***R***のバッグとし，挿入操作 ***i***: **insert into** V **values** I;が発行されたとすると，3.2.2項の議論と全く同様に***i***の変換には曖昧性が生じるので，変換不可である．この性質はすべてのR, S, Iについて成り立つからスキーマレベルである．

3.3 バッグ意味論の下でのクロス結合ビューの更新可能性

バッグ意味論の下でのクロス結合ビューの更新可能性は，2.6.2項で論じたが，場合により削除可能，挿入可能，そして書換可能である．この更新可能性はインスタンスレベルである．

3.4 バッグ意味論の下での射影ビューの更新可能性

射影ビューV=R[X]の定義より次が成り立つ．

∀R∀t∀k(t(k)∈V iff ∃u(u(k)∈R∧t=u[X]))

集合意味論の下では射影はテーブルRからV，ここに$X\subseteq\Omega_R$への一般に多対1の縮退写像であるが，バッグ意味論の下では上記の定義によりバッグRからVへの1対1の全単射である．従って，この写像の下で射影ビューは削除可能である．射影ビューへの挿入は，t(k)を挿入したい行としたとき，tを$\forall A\in(\Omega_R-\Omega_{R[X]})$については"空"(null)で補完した行を$t^c$とすれば，$t^c(k)$をRに挿入する操作に変換できる[11]．これらの変換はスキーマレベルである．

3.5 バッグ意味論の下でのθ-選択ビューの更新可能性

$V=R[A_i\,\theta\,A_j]=\{t_i(k_i)\,|\,t_i\in\delta(R)\wedge t_i[A_i]\,\theta\,t_i[A_j]\}$を一般にバッグ $R=\{t_1(k_1), t_2(k_2), ..., t_p(k_p)\}$の$\theta$-比較可能な属性 A_i と A_j 上

[10] ビューVに対する書換操作***r***はまず削除操作***d***を行い，続いて挿入操作***i***を行うという系列と見なされる．このとき，***d***について意図に基づくアプローチの下でその変換可能性を検証するとT(***d***)と変換可能であったとする．続いて，***r***を実現させるためには，***d***を施されたVに対して挿入操作***i***が変換可能でなければならないが，これもT'(***i***)と変換であったとしよう．このとき，「変換T(***d***)とT'(***i***)は書換操作***r***に関して両立(compatible)している」，略して「書換操作***r***が両立している」という．

[11] この操作がRの制約に抵触するか否かは別問題である．

のθ-選択ビューとする．定義より次が成り立つ．

$\forall R \forall t \forall k(t(k) \in V$ iff $(t(k) \in R \wedge t[A_i]\,\theta\, t[A_j]))$

従って，V への挿入操作 **insert into** V **values** {t(k)};ここに $t[A_i]\,\theta\, t[A_j]$が成立，は R への挿入操作 **insert into** R **values** {t(k)};に一意で副作用なく変換可能である．V への削除操作は **delete** D **from** V;は同種変換の原理[12]の下で **delete** D **from** R;に一意に副作用なく変換可能である．これらはスキーマレベルの性質である．なお，同種変換の原理の詳細は文献[7]を参照されたい．

3.6 バッグ意味論の下でのθ-結合ビューの更新可能性

―一般的に定義されるビューの更新可能性検証―

一般にビューはバッグ代数表現として再帰的に定義されるが(定義 2)，そのようなビューの更新可能性の検証はそのビューを定義するビュー定義木に従ってトップダウンで行われる．例えば，バッグ R と S のθ-比較可能な属性 $R.A_i$ と $S.B_j$ 上のθ-結合ビュー$V=R[A_i\,\theta\,B_j]S=(R\times S)[R.A_i\,\theta\,S.B_j]$に対して，削除操作 ***d*** が発行されたとすると，中間ビューを $V^{int}=R\times S$ とし，$V=V^{int}[R.A_i\,\theta\,S.B_j]$に対して ***d*** が発行されたとして，その更新可能性がまず問われる．バッグ意味論の下でのθ-選択ビューは同種変換の原理の下で削除可能であるので(3.5 節)，***d*** は V^{int} への削除操作 ***d'***に変換され，次はその削除可能性を問う問題に帰着される．クロス結合ビューは場合により削除可能であるから，結果的に．θ-結合ビューは同種変換の原理の下で場合により削除可能である．挿入操作の変換可能性も同様に再帰的に検証され，場合により挿入可能となる．従って，θ-結合ビューは同種変換の原理の下で場合により書換可能である．これらはインスタンスレベルの性質である．

以上の結果をまとめて表 1 に示す．クロス結合ビューとθ-結合ビューの更新可能性はインスタンスレベル，その他はスキーマレベルアプローチで規定されている．和―削除，差―挿入，共通―挿入の組合せについては集合意味論の下では変換可能である(＊印)．また，射影―挿入，射影―書換については集合意味論の下で変換可能であるがキー保存という条件がつく(★印)[7]．

3.7 HISU; Hybrid Instance-level/Schema-level Updatability

一般にインスタンスレベルアプローチはスキーマレベルアプローチと比べて，更新可能性を検証するためにビューを一旦マテリアライズしないといけないから処理コストが嵩むが，スキーマレベルでは更新不可であったクロス結合ビュー(とθ-結合ビュー)が場合により更新可能となるという利点がある．これは集合意味論の下でもバッグ意味論の下でも成り立つ，

一方，前節で示したように，重複行削除，和，差，共通，射影，θ-選択ビューの更新可能性はスキーマレベルで規定される．スキーマレベルアプローチの利点はそのコスト安にある．すなわち，このアプローチの下では，ビューの更新可能性はその定義とそれに対して発行された更新操作のタイプ[13]で判定できるから，その更新可能性を“メタデータ”として格納することにより，ビューに対して更新操作が発行された時点でその更新可能性をテーブル参照(table lookup)で決定できるのでコスト安と考えられる．

表 1 バッグ意味論の下での基本ビューの更新可能性

	重複行除去	和	差	共通	クロス結合	射影	θ-選択	θ-結合
D	◯	×*	×	×	◎	◯	◯†2	◎†2
I	×	×	×*	×*	◎	◯★	◯	◎
R	×	×	×	×	◎†1	◯★	◯†2	◎†1,2

D：削除，I：挿入，R：書換．◯：変換可能(スキーマレベル)，◎：変換可能(インスタンスレベルで可能．スキーマレベルでは不可)，×：変換不可(和―D では $m\neq 0 \wedge n\neq 0$ の条件付き)．†1：書換操作の両立，†2：同種変換の原理（†1～†2 の詳細は[7]を参照）．

そこで，クロス結合ビュー(とθ-結合ビュー)についてはインスタンスレベルアプローチ，すなわち意図に基づくアプローチで，その他は従来のスキーマレベルアプローチでビューに対する更新操作の変換可能性を検証する手法が考えられる．これを HISU（Hybrid Instance-level/Schema-level Updatability, 混合インスタンスレベル／スキーマレベル更新可能性)と名付ける．本論文はこの枠組みでの議論である．

先に進む前に，ここで意図に基づくアプローチの妥当性について補足する．このアプローチに対して，「ユーザが必ずしも全てを把握していない，データベースのその時点のインスタンスにビュー更新の変換可能性が依存してしまうことになるから，これで本当にユーザの意図を把握したことになるのか?」との疑念を耳にすることがある．これはもっともに聞こえる．しかし，「では，この疑念は（インスタンスレベルではなく)スキーマレベルアプローチを採れば解消できるのですか？」と反問すれば，実はそこでも同様な疑念が生じて，上述のような疑念は何も意図に基づくアプローチだけに向けられるべきものでもないことに気が付く．即ち，スキーマレベルアプローチでは全ての時点のインスタンスと全ての同じタイプの更新操作の組合せで定義 3 に与えられた変換可能性の成立が問われることになるが，「では，このアプローチで変換可能とされたビューの更新変換は本当にユーザの意図を把握したことになりますか？」と問われれば，「いや，そうとは言えない」が正答であろう．何故ならば，ユーザの更新意図はユーザに聞くしかないからである．実際，この観点から対話的ビューサポートシステム TAILOR[10]の開発が報告されている．ただ，両アプローチの差異が両極にあることには注意する[14]．そもそも，これまで広く受け入れられているビューの更新可能性の定義 3 は Dayal ら[2]によるが，その定義はインスタンスレベルで与えられている．この定義を基に，ビューの更新可能性をスキーマレベルで捉えるか，インスタンスレベルで捉えるか，そこには裁量が入り込む余地が残されていた．そして，これまでは理論でも実践でもビューの更新可能性はスキーマレベルアプローチの下で論じられ実践されてきた．その理由，そして意図に基づくアプローチの意図するところは上述の通りである．

---

[12] 例えば，リレーション社員(社員名，給与）の<-選択ビュー(＝小なり選択ビュー）貧乏社員=社員[給与<20] を定義し，それに対して削除操作 ***d***：delete（高橋次郎, 15）from 貧乏社員;が発行されたとする．この削除操作の変換候補は，(a) T_1(***d***)：delete（高橋次郎，15）from 社員;，そうでなければ，(b) 書換操作，例えば T_2(***d***)：rewrite（高橋次郎, 15）of 社員 to（高橋次郎, 30); の 2 通りが考えられる．従って，この曖昧性から<-選択ビューは削除不可と結論される．しかしながら，もしθ-選択ビュー$V=R[Ai\,\theta\,Aj]$に対して発行された削除操作は，R に対する挿入操作や書換操作ではなく，また削除操作に変換されなければならないという規則を導入すると，上記の ***d*** の変換候補は T_1(***d***)のみと唯一となり変換可能となる．これを同種変換の原理という．

[13] 削除操作なのか挿入操作なのか書換操作なのか，ということ．

[14] つまり，ビューの更新可能性について，インスタンスレベルアプローチでは定義 3 を満たすデータベースのインスタンスと更新操作の組合せの“存在”(∃)が問われ，一方，スキーマレベルアプローチではそれら“全て”(∀)の組合せにおいて定義 3 が満たされているか否かが問われている．そこに両極（∃か∀）ともいえる差異がある．

4. 整合ラベリング問題としてのクロス結合ビューの更新可能性

4.1 非線形連立方程式としての形式化

ビュー更新問題を制約充足問題(CSP)と捉える研究を Shu[11] に見ることができるが，そこではビューの更新問題が2項CSPとして形式化できることが報告されている．一方，我々の目的はクロス結合ビューへの更新操作の変換可能性検証について，これまでの力まかせ法に代わる手法を模索した結果，それがCSPの一つである整合ラベリング問題(CLP)に帰着できることを示すことで，CLP 法と名付けた新手法を提案することにある．そのために，まず，クロス結合ビューの更新問題が非線形連立方程式を解く問題に帰着できることを示すことから始める．

そこで，一般に R(A)と S(B)，ここに A と B はリレーションスキーマ $\boldsymbol{R}$ と $\boldsymbol{S}$ の全属性集合を表すとする，をバッグとし，V=R×S をそれらのクロス結合ビューとする．このとき，V に対して削除操作 $\boldsymbol{d}$: **delete** D **from** V; ここに，D はリレーションスキーマ $\boldsymbol{V}$ のバッグとする，が発行されたとする．V から D の削除が実現されるためには，R から何本かの行が削除され，S からも何本かの行が削除される必要があろう．しかし，R や S のどの行が何本削除されるべきなのかは，D が与えられただけで直ちに分かるわけではない．そこで，削除や挿入に関与すると考えられる R と S の行の重複度を変数（x_i や y_j）で表し，これを重複度変数と呼ぶ，かつそれらのドメイン(dom)を求めて，D をそれら全体が満たすべき制約と捉えることにより，重複度変数 x_i と y_j が満たすべき非線形連立方程式を形成し，それが唯一解を持つとき及びそのときのみ $\boldsymbol{d}$ は変換可能となる，とクロス結合ビューの更新問題を形式化する．力まかせ法で更新可能性を検証した例題2と同じ例を用いて説明する．

【例題4】R(A)={1(1), 2(2)}, S(B)={10(1), 20(3)}を実バッグ，R と S のクロス結合ビューを V=R(A)×S(B)とする．V に対して削除操作 $\boldsymbol{d}$: **delete** D **from** V; ここに, D={(2, 10)(2), (2, 20)(6)}とする，が発行されたとする．すると，δ(D[A])∩δ(R)={2}, δ(D[B])∩δ(S)={10, 20}なので，重複度変数 x_1, y_1, y_2 を導入し，R^{new}(A)={1(1), 2(x_1)}，S^{new}(B)={10(y_1), 20(y_2)}とする．このとき dom(x_1)={0, 1, 2}, dom(y_1)={0, 1}, dom(y_2)={0, 1, 2, 3}である．そうすると，R^{new}(A)×S^{new}(B)={(1, 10)(1×y_1), (1, 20)(1×y_2), (2, 10)(x_1×y_1), (2, 20)(x_1×y_2)}であり，一方，V−D={(1, 10)(1), (1, 20)(3)}であり，そして，R^{new}(A)×S^{new}(B)=V−D でなければならないから，$\boldsymbol{d}$の変換可能性は次に示す連立方程式が唯一の解を有するか否かという問題に帰着される．

dom(x_1)={0, 1, 2}, dom(y_1)={0, 1},
dom(y_2)={0, 1, 2, 3}・・・(0)
1×y_1=1・・・(1)　　1×y_2=3・・・(2)
x_1×y_1=0・・・(3)　　x_1×y_2=0・・・(4)

更に，この連立方程式の線形関係(1)，(2)を(0)式に適用して，次に示す非線形連立方程式が得られる．

dom(x_1)={0, 1, 2}, dom(y_1)={1}, dom(y_2)={3}・・・(0)
x_1×y_1=0・・・(1)　　x_1×y_2=0・・・(2)

この非線形連立方程式の解は x_1=0, y_1=1, y_2=3 と唯一で，よって $\boldsymbol{d}$ は **delete** {2(2)} **from** R; に変換されねばならないことが R, S と R^{new}, S^{new} の対応する行の重複度の比較で分かる．

一方，クロス結合ビューへの挿入操作の変換可能性も次に示すように非線形連立方程式へと帰着できる．例題3と同じ例を用いて説明する．

【例題5】R(A)={1(1), 2(2)}, S(B)={10(1), 20(3)}を実バッグ，R と S のクロス結合ビューを V=R(A)×S(B)とする．V に対して挿入操作が発行されたとする：$\boldsymbol{i}$: **insert into** V **values** I; ここに，I={(1, 30)(2), (2, 30)(4)}とする．

すると，δ(I[A])={1, 2}, δ(I[B])={30}，30∉δ(S)なので，下に示す非線形連立方程式が得られる．ここに，重複度変数 y_1 は S に挿入される行30の重複度を表し，R^{new}(A)={1(x_1), 2(x_2)}, S^{new}(B)={10(1), 20(3), 30(y_1)}，dom(x_1)={1, 2, 3}, dom(x_2)={2, 3, 4, 5, 6}, dom(y_1)={1, 2, 3, 4, 5, 6}である．また，V をマテリアライズして，V⊎I={(1,10)(1), (1,20)(3), (1, 30)(2), (2,10)(2), (2,20)(6), (2, 30)(4)}となる．一方，R^{new}(A)×S^{new}(B)={(1, 10)(x_1×1), (1, 20)(x_1×3), (1, 30)(x_1×y_1), (2, 10)(x_2×1), (2, 20)(x_2×3), (2, 30)(x_2×y_1)}．従って，次に示す連立方程式が得られる．

dom(x_1)={1, 2, 3}, dom(x_2)={2, 3, 4, 5, 6},
dom(y_1)={1, 2, 3, 4, 5, 6}・・・(0)
x_1×1=1・・・(1)　　x_1×3=3・・・(2)
x_1×y_1=2・・・(3)　　x_2×1=2・・・(4)
x_2×3=6・・・(5)　　x_2×y_1=4・・・(6)

線形関係(1)，(2)，(4)，(5) を(0)式に適用して，次の非線形連立方程式を得る．

dom(x_1)={1}, dom(x_2)={2}, dom(y_1)={1, 2, 3, 4, 5, 6}・・・(0)
x_1×y_1=2・・・(1)　　x_2×y_1=4・・・(2)

この非線形連立方程式の解は x_1=1, x_2=2, y_1=2 と唯一で，よって $\boldsymbol{i}$ は **insert into** S **values** {30(2)}; に変換されればよいことが分かる．

なお，言うまでもないが，力まかせ法で得た解（例題2，同3)とこの非線形連立方程式を解いて得た解(例題4，同5)は一致している．

4.2 整合ラベリング問題としての形式化

さて，クロス結合ビューの更新問題は非線形連立方程式を解く問題に帰着されたが，これは制約充足問題(constraint satisfaction problem, CSP)[12]の一種である整合ラベリング問題(consistent labeling problem, CLP)となっている．我々は CLP の解法として西原[8]により提案された併合法(merge method)に着目した．その理由はこの手法が全解(all solutions)を求めるための手法であるからである．これが我々の目的に合致している．なぜならば，もし複数の解が求まったとするならばそれはビューに対して発行された更新操作の変換に一意性がなく変換に曖昧性が生じることを，もし解が無ければそもそも変換候補が存在せず変換はできないことを，そして単解のときにのみ変換が可能であることを意味しているからである．

そこで，CLP の定義を与え，問題を形式化する．

【定義4】CLP は4項組(U, L, T, R)である．ここに，U, L, T, R はそれぞれ，ユニットの集合，ラベルの集合，ラベル間の制約の集合，そして局所解(local solution)の集合である．

そこで，例題4を CLP として表現すると次のようである：

U={x_1, y_1, y_2}　　L={0, 1, 2, 3}
T={t_1, t_2}，ここに t_1=(x_1, y_1)，t_2=(x_1, y_2).
R={R_1, R_2}，ここに R_1={(0, 1)}, R_2={(0, 3)}.

各 R_i の元の数を R_i のサイズという．2項関係(t_i, R_i)を制約対，V_i=(t_i, R_i)を頂点という．

さて，併合法の動作は，制約ネットワーク(constraint network)を導入すると分かり易い．上で示した例題4のU，L，T，Rを例としてそれを描くと図3(a)の通りとなる(この例では，直線グラフになったが，一般には多数の頂点からなる結合グラフとなる)．

併合法のストラテジに基づき，図3(a)に示された制約ネットワークの2つの頂点V_1とV_2を併合(merge)すると，同図(b)に示された併合結果を得る．

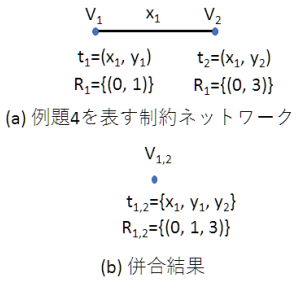

図3 例題4を表す制約ネットワークと併合結果

4.3 併合結果の解釈と変換可能性

上記の結果でまず着目するべきは，$R_{1,2}$がただ一つの元(0, 1, 3)から成り立っている単集合(singleton)であることである．これは例題4で示したビューV=R×Sに対する削除操作 ***d*** が一意に副作用なく変換可能であることを意味している．

では，このとき，実際にビューVに対する削除操作 ***d*** は実バッグRとSへのどのような更新操作に変換されるのであろうか，その算出法を次に示す．

① (0, 1, 3)はx_1=0, y_1=1, y_2=3を意味する．

② 一方，x_1, y_1, y_2の初期値はx_1=2, y_1=1, y_2=3であった．

③ 両者を比較すると，x_1の値が初期値2から0に変化していることが分かる(他は変化していない)．

④ ところで，x_1はバッグR(A)中のA値が2の行数を表していた．つまり，この例では，最初はRにA値が2の行が2本あったが，***d*** を実現するためにはそれらを全て削除して0本にするべきだといっている．

⑤ 従って，***d*** は次の操作に変換可能である：

T(***d***): **delete** {2(2)} **from** R;

実際に，T(***d***)をRに適用して，その結果とSをクロス結合すれば，所望のビュー更新結果が得られることを確認できる．

続けて，クロス結合ビューに対する削除操作が，解がないために変換不可と結論される場合と，解が複数あるが故に変換不可とされてしまう例を示して上記の議論を補う．

【例題6】R(A)，S(B)，それらのクロス結合ビューVは例題4で与えた通りとする．Vに対して次の削除操作が発行されたとする：***d'***: **delete** D **from** V; ここに，D={(1, 10)(1), (2, 20)(6)}とする．δ(D[A])={1, 2}，δ(D[B])={10, 20}なので，重複度変数x_1, x_2, y_1, y_2を導入し，R^{new}(A)={1(x_1), 2(x_2)}，S^{new}(B)={10(y_1), 20(y_2)}とする．このとき dom(x_1)={0, 1}，dom(x_2)={0, 1, 2}，dom(y_1)={0, 1}，dom(y_2)={0, 1, 2, 3}である．そうすると，R^{new}(A)×S^{new}(B)={(1, 10)(x_1×y_1)，(1, 20)(x_1×y_2)，(2, 10)(x_2×y_1)，(2, 20)(x_2×y_2)}であり，一方でV−D={(1, 20)(3), (2, 10)(2)}はR^{new}(A)×S^{new}(B)と等しくなければならない．従って，***d'***の変換可能性は次に示す非線形連立方程式が唯一の解を有するか否かという問題に帰着されたことが分かる．

dom(x_1)={0, 1}, dom(x_2)={0, 1, 2}, dom(y_1)={0, 1},
dom(y_2)={0, 1, 2, 3}・・・(0)

x_1×y_1=0・・・(1)　　x_1×y_2=3・・・(2)

x_2×y_1=2・・・(3)　　x_2×y_2=0・・・(4)

この非線形連立方程式の解を求めるために，問題をCLPとして表現すると，U={x_1, x_2, y_1, y_2}，L={0, 1, 2, 3}，T={t_1, t_2, t_3, t_4}，ここにt_1=(x_1, y_1)，t_2=(x_1, y_2)，t_3=(x_2, y_1)，t_4=(x_2, y_2)，R={R_1, R_2, R_1, R_2}，ここにR_1={(0, 0), (0, 1), (1, 0)}，R_2={(1, 3)}，R_3={(2, 1)}，R_4={(0, 0), (0, 3), (2, 0)}となる．この場合の制約ネットワークは図4に示される通りである．併合法のストラテジに従って頂点を順次併合していくと，結果は$R_{1,2,3,4}$=∅となり．解は存在しないことが分かる．依って，***d'***は変換不可である．

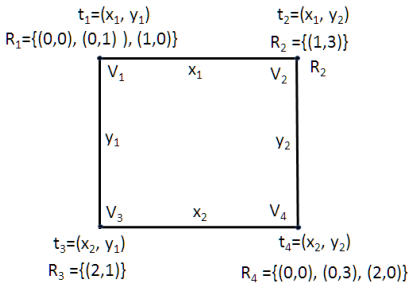

図4　例題6の制約ネットワーク

次に，併合結果が複数の元からなることで変換の曖昧性が生じ，従って変換不可となるという例を，例題1で与えたバッグとビューを用いて示す：バッグをR(A)={1(2)}，S(B)={10(2)}，それらのクロス結合ビューをV=R×Sとし，Vに対して削除操作 ***d''***: **delete** {(1, 10)(2)} **from** V; が発行されたとする．重複度変数x_1とy_1を導入し，R^{new}(A)={1(x_1)}，S^{new}(B)={10(y_1)}とすると，Vをマテリアライズして，(V−{(1, 10)(2)})={(1, 10)(2)}となるから，x_1×y_1=2，ここに dom(x_1)=dom(y_1)={0, 1, 2}である．従って，その制約ネットワークは1頂点からなり，その局所解はR={(1, 2), (2, 1)}となり，単解ではない．実際，これは ***d''***を **delete** 1(1) **from** R; あるいは **delete** 10(1) **from** S; の何れに変換しても実現できることを示している．従って，変換の曖昧性により ***d''***は変換不可と結論される．

ここで，クロス結合ビューへの挿入操作の変換可能性も整合ラベリング問題に帰着されることを例題5を用いて示しておく．そこでは，クロス結合ビューV=R(A)×S(B)に対して発行された挿入操作 ***i***: **insert into** V **values** I; ここに，I={(1, 30)(2), (2, 30)(4)}の変換可能性は次に示す非線形連立方程式が唯一解を有するか否かと規定された(分かり易さのために再掲)．

dom(x_1)={1}, dom(x_2)={2}, dom(y_1)={1, 2, 3, 4, 5, 6}・・・(0)

x_1×y_1=2・・・(1)　　x_2×y_1=4・・・(2)

この制約ネットワークは2頂点V_1=(t_1, R_1), V_2=(t_2, R_2) からなる結合グラフである，ここにt_1=(x_1, y_1)，t_2=(x_2, y_1)，R_1={(1, 2)}，R_2={(2, 2)}．併合法のストラテジに則り2頂点を併合すると，局所解$R_{1,2}$={(1, 2, 2)}が得られる．これはx_1=1, x_2=2, y_1=2を意味するから，***i*** は **insert into** S **values** {30(2)}; に変換されればよいことが分かる．

以上，クロス結合ビューへの更新操作の変換可能性を整合ラベリング問題として検証する手法を「CLP法」と呼ぶことにする．

4.4 局所解のSQL２項テーブル表現

CLP 法での局所解を SQL の２項テーブルとして表現することができ，それにより制約ネットワークの頂点の併合操作を 2 項テーブルの自然結合(natural join)をとる操作として実装できることを示す．例えば，例題 6 では局所解は R_1={(0, 0), (0, 1), (1, 0)}, R_2={(1, 3)}, R_3={(2, 1)},R_4={(0, 0), (0, 3), (2, 0)}であった．このとき，R_1 は重複度変数 x_1 と y_1 が満たすべき局所解を表していることに注意すると(R_2~R_4 についても同様で，R_2 は(x_1, y_2)，R_3 は(x_2, y_1)，R_4 は(x_2, y_2)が満たすべき局所解を表している)，それらは図 5 に示されるような SQL の２項テーブルとして表現することができる．これらを「局所解テーブル」と呼ぶこととすると，併合法で制約ネットワークの頂点を併合していく操作は，これら 4 つの局所解テーブルの自然結合 R_1*R_2*R_3*R_4 をとることに等しい（この例では，導出表は∅となる)．

R_1

x_1	y_1
0	0
0	1
1	0

R_2

x_1	y_2
1	3

R_3

x_2	y_1
2	1

R_4

x_2	y_2
0	0
0	3
2	0

図5 例題6の局所解のSQLの２項テーブル表現

併合法を用いることで，クロス結合ビューに対する更新操作の変換可能性を検証できることは 4.2 節で述べたところであるが，ここで，制約ネットワークの頂点の併合操作を 2 項テーブルの自然結合をとる操作として実装できることの意義を再確認しておく．１つは，クロス結合ビューに対する更新操作の更新可能性を“リレーショナルDBMSの問合せ処理機能をそのまま使用して検証する道を拓いた”ということである．すなわち，すべての局所解の２項テーブルの“自然結合”をとった結果得られた導出表が，1 タップルからなれば更新操作は変換可能，空集合になれば変換が存在しないという意味で変換不可，2 タップル以上になれば変換の曖昧性があるということで変換不可，と規定できることにある．もう１つは，併合法では制約ネットワークの２つの頂点を併合していくときに，どの２頂点から併合していくと併合に係るコストを極小にできるかを選択するためにコスト推定を行うことが必要となるが[8]，併合操作が局所解テーブルの自然結合演算に置き換わると，この選択問題はリレーショナル DBMS の最適化器(optimizer)の所掌となる．つまり，併合法をそのまま実装しようとすればコスト推定プログラムを開発し実行させる必要があるが，局所解を 2 項テーブル表現することで選択問題をリレーショナルDBMSが解いてくれることになり，不必要となる．CLP 法を局所解の SQL の２項テーブル表現を用いて実装した場合の計算量は 5.1.2 節で論じる．

5. 考察

5.1 力まかせ法とCLP法の計算量

バッグR(A)とS(B)のクロス結合ビューV= R(A)×S(B)への更新操作の変換可能性を力まかせ法と CLP 法で検証する場合の計算量の違いを考察する．なお，R(A)，S(B)，D，I は重複度付き行表現標準形で与えられ，V はマテリアライズされているとする．また，両手法共に，ビューの更新可能性検証実行の際に必要なリレーショナル代数演算やバッグ代数演算に係る処理はリレーショナル DBMS に実行させることとし，そのための計算量を“リレーショナルDBMS 依存である”と定義する（この定義の意味するところは 5.1.3 節で記す)．

5.1.1 力まかせ法の計算量

まず，削除操作 ***d***: **delete** D **from** V; が発行されたとする．ここに，D={$u_1(k_1)$, $u_2(k_2)$, ..., $u_p(k_p)$}とする．力まかせ法の手順は 2.6.2 節で示した通りである．従って， ***d***の変換可能性を検証するための計算量を $C_{BF\text{-}d}$とすると次のようである．

$C_{BF\text{-}d}$= δ(D[A])∩δ(R)と δ(D[B])∩δ(S)を求めるための計算量①＋R^dと S^dの全候補を求めるための計算量②＋
(V−D)=(R−R^d)×(S−S^d)の成立を検証するための計算量③

計算量①は射影，重複行除去，共通演算に係る計算量でリレーショナル DBMS 依存である．計算量②は δ(D[A])∩δ(R)={$t_{1'}$, $t_{2'}$, ..., $t_{q'}$}ならば，$t_{i'}(k_{i'})\in R$ として，R^d の総候補数は $\Pi_{i=1,\ ...,\ q}(k_{i'}+1)$となる．そこで，$(k_{1'}+1)$～$(k_{q'}+1)$の最小値を m とすれば，$m^q \leqq \Pi_{i=1,\ ...,\ q}(k_{i'}+1)$であるから，$R^d$の総候補数を求める計算量は指数時間である．$S^d$についても同様である．計算量③は R^dと S^d の候補すべての組合せについて検証しないといけないので，その計算量は指数時間となる．

V に対して挿入操作 ***i***: **insert into** V **values** I; が発行された場合の計算量は，検証手順の違いから削除の場合とは若干異なるが，同様の導出となる．

5.1.2 CLP法の計算量

まず，削除操作 ***d***(力まかせ法で与えたものと同じ)の変換可能性を検証するための計算量を $C_{CLP\text{-}d}$とすると次のようである．

$C_{CLP\text{-}d}$=(a) δ(D[A])と δ(D[B])を求め，(b) その結果に基づき重複度変数並びにそれらのドメインを求め，(c) R^{new}(A)と S^{new}(B)を生成するための計算量①＋
$R^{new}(A)\times S^{new}(B)=V-D$ が成立しなければならないという条件から非線形連立方程式を導出するための計算量②＋
局所解を求めて局所解テーブルを作成し，それらの自然結合をとり変換可能性を検証するための計算量③

計算量①-(a)に係る計算量はリレーショナル DBMS 依存，①-(b)に係る計算量はすべてが突合せ処理なので多項式時間，①-(c)に係る計算量も同様に多項式時間である．計算量②は，$R^{new}(A)\times S^{new}(B)$と V−D を導出し，対応した行の重複度を比較して連立方程式を求め，更に線形関係をドメインに適用して，結果として非線形連立方程式を求める処理で，基本的に突合せ処理なので，多項式時間である．補足すれば，対(x_i, y_j)の総個数は高々|δ(R)|×|δ(S)|である．計算量③は，まず局所解を求めるが，$x_i\times y_j=k_{i,j}$ であったとすれば，dom(x_i)×dom(y_j)から $x_i\times y_j=k_{i,j}$ となる元を抽出すると局所解が求められるから，|dom(x_i)|あるいは|dom(y_j)|の最大値を m とすれば，全ての局所解を求めるための計算量は，$m^2\times|\delta(R)|\times|\delta(S)|$と多項式時間である．局所解を局所解テーブルに変換する計算量は多項式時間である．これにより併合処理はすべての局所解テーブルの自然結合をとることで行え，その導出表を用いて判定できることとなる(4.4 節)．以上から，$C_{CLP\text{-}d}$はリレーショナル DBMS 依存である．

挿入の場合，議論は削除の場合とほぼ同様であるが，ドメインの濃度の最大値を m とすれば，局所解を求める計算量は $m^2\times|\delta(R\uplus I[A])|\times|\delta(S\uplus I[B])|$と多項式時間であるが，削除の場合と同様に，局所解テーブルを求め，それらを自然結合してその導出表を検証することで挿入操作の変換可能性を検証するので，計算量 $C_{CLP\text{-}i}$はリレーショナル DBMS 依存である．

5.1.3 計算量がリレーショナルDBMS依存という定義

リレーショナル代数演算やバッグ代数演算の処理をリレーショナル DBMS に任せたとき，その処理の計算量を“リレーショナル DBMS 依存である”と定義したことについて，まず例をあげて補足する．1 例目はバッグ R に重複行削除演算δを施し，リレーションδ(R)を求める処理の計算量である．このために，リレーショナル DBMS はまず R をソート(sort)し，続いて重複行を削除するのが一般的であろうから，例えばクイックソート法でその処理が行われたとすれば，その平均計算量は|R|×log|R|である．2 例目は自然結合演算の計算量である．n 個のテーブルの自然結合$R=R_1*R_2*...*R_n$をナイーブに実行すれば，それらのテーブルの濃度の最小値を m とすれば，少なくとも m^n 回のタップル結合処理を行わないといけないから，この計算量は結合するテーブル数の指数オーダとなる．従って，理論上，自然結合演算では組合せ爆発が起こり，有限時間で解を求めることが困難という性質を有する．しかし，自然結合演算を司るリレーショナルDBMSの質問処理系は，だからと言ってその処理を放棄はしない．n や m がそこそこの大きさであればそれなりの時間で結果を返せるだろうし，勿論，自然結合の計算量は指数時間であることは承知の上で，最適化器は出来うる限り効率の良い結合法でそれを処理してユーザの要求に応える．理論的な計算量は CPU 能力を向上させたからといって下がるものでもない．アルゴリズムの実装にリレーショナル DBMS を取り込んだ場合，そこでの計算量は何を処理するかに依存して多様ではあるものの，包括的に「リレーショナル DBMS は入力に対して出力を返してくれる関数である」と捉えて，その計算量を “リレーショナル DBMS 依存”と定義したということである．

5.2 マテリアライズドビューの更新可能性

これまでマテリアライズドビューのサポートについては，マテリアライズドビューを定義するために使われた実テーブルに更新があった場合にその変化分を如何に効率よくマテリアライズドビューに反映させるかという増分ビューメンテナンス（incremental view maintenance, IVM)の研究・開発に力点が置かれてきた[4]．換言すると，これまでマテリアライズドビューは読取り専用との位置づけが定着化しており，従ってマテリアライズドビューの更新可能性については議論がなされてこなかった．しかしながら，意図に基づくアプローチと CLP 法はインスタンスレベルでビューの更新可能性を検証するので，マテリアライズドビューの更新問題にそのまま適用可能と考えられる．つまり，意図に基づくアプローチでビューの更新可能性を検証するには，当該ビューをマテリアライズしないといけないから，そのために計算資源を食うという負の側面が指摘されるが，マテリアライズドビューの更新可能性を検証する場合にはその問題は発生せず，本論文で提案した手法をそのまま適用できる．近年，マテリアライズドビューのビジネスインテリジェンス(BI)のためのオンライン分析処理（OLAP）への提供が加速している．そこではマテリアライズドビューを読取り専用ではなく，更新の対象ともしたいという要求がある．その要求に応えられるのが意図の基づくアプローチである．

5.3 ビュー更新とトリガ

SQL:1999 でトリガ(trigger)が標準化された[5]．トリガはテーブルの更新を引き起こす特定のデータ操作を契機として，あらかじめデータベースに登録した SQL 手続きを起動する機能である．トリガはデータベースの一貫性を維持するためによく用いられるが，テーブルがビューである場合も定義可能である．そのためであろうが，“トリガを使うとビューは自由に更新できる”と謳う記事を Web 上で散見する．しかし，その目的でトリガを安易に用いることは大変危険である．なぜならば，ビューの更新可能性に留意しないで書き下したトリガはデータベースの一貫性を容易に破壊するであろうからである．SQL の現場で更新を許される結合ビューは極めて限定的であることは第 1 章で述べた通りであり，結合ビューをトリガを使って更新できればと欲するのは尤もなことである．その際，意図に基づくアプローチは，“正しい”，つまりデータベースの一貫性を損なうことのないトリガのコアコードを書き下すための指針となるであろう．

6. おわりに

意図に基づくアプローチを SQL が準拠するバッグ意味論の下に拡張し，そのビューサポート能力を明らかにした．ビューサポートがインスタンスレベルで規定されているのか，スキーマレベルで規定されているのかを論じ，HISU と名付けられた新しいビューサポートの枠組みを提案した．さらに，バッグ意味論の下でのクロス結合ビューの更新可能性が整合ラベリング問題(CLP)に帰着でき，その解法の一つとして知られている併合法を用いて解けることを示した．CLP 法と名付けらたこの新規アプローチは従来の力まかせ法と比較して SQL 環境での実装に親和性がある．この新方式が HISU の枠組みの下で，今後商用やオープンソースのリレーショナル DBMS のビュー(マテリアライズドビューを含む)サポート機能の強化・刷新の指針となることを期待する．

末筆ながら，本稿をまとめるにあたり，数々の有益なコメントを下さった匿名の査読者に謝意を表する．

参考文献

[1] Wayne D. Blizard. Multiset Theory. *Notre Dame Journal of Formal Logic,* Vol.30, No.1, pp.36-66, 1989.
[2] Umeshwar Dayal and Philip A. Bernstein. On the Updatability of Relational Views. *Proc. of the 4th VLDB,* pp.368-377, 1978.
[3] Paul W. P. J. Grefen and Rolf de By. A Multi-Set Extended Relational Algebra – A Formal Approach to a Practical Issue. *Proc. of the 10th Intl. Conf. on Data Engineering*, pp.80-88, 1994.
[4] Ashish Gupta and Inderpal Singh Mumick (ed.). Materialized Views – Techniques, Implementations, and Applications. *The MIT Press*, 589p., 1999.
[5] ISO/IEC 9075-2:1999 Information technology -- Database languages -- SQL -- Part 2: Foundation (SQL/Foundation).
[6] Yoshifumi Masunaga. A Relational Database View Update Translation Mechanism. *Proc. of the 10th VLDB,* pp.309-320, 1984.
[7] 増永良文，長田悠吾，石井達夫．更新意図の外形的推測に基づいたビューの更新可能性とその PostgreSQL 上での実現可能性検証．日本データベース学会和文論文誌，Vol. 18-J, Article No. 1, 2020 年 3 月.
[8] 西原清一．整合ラベリング問題と応用．情報処理, Vol. 31, No. 4, pp.500-507, 1990.
[9] Oracle Database Administrator's Guide, Updating a Join View. https://docs.oracle.com/cd/B28359_01/server.111/b28310/views001.htm#ADMIN11782.
[10] Amit P. Sheth, James A. Larson and Evan Watkins. TAILOR, A Tool for Updating Views. *LNCS*, Vol. 303, pp.190-213, Springer, 1988.
[11] Hua Shu. Using Constraint Satisfaction for View Update Translation. *Proc. of the 13th European Conference on Artificial Intelligence*, pp.33-37, 1998.
[12] Edward Tsang. Foundations of Constraint Satisfaction. *Herstellung und Verlag: BoD*, Norderstedt, Germany, 421p., 1996.