バッグ意味論のもとでのビュー更新問題の検討
－更新意図の外形的推測に基づくアプローチの適用可能性－

増永 良文† 長田 悠吾‡ 石井 達夫‡

†お茶の水女子大学 〒112-8610 東京都文京区大塚 2-1-1
‡ SRA OSS, Inc. 日本支社 〒171-0022 東京都豊島区南池袋 2-32-8
E-mail: †yoshi.masunaga@gmail.com, ‡{nagata, ishii}@sraoss.co.jp

あらまし リレーショナルデータベースのビュー更新問題に対して，我々は，従来のビューサポートの限界を打ち破るべく，「更新意図の外形的推測に基づくアプローチ」（簡単に，意図に基づくアプローチという）という新しい考え方を提案して，その有効性を検討してきた．この理論は純粋にリレーショナルデータモデルに立脚して展開されており，いわゆる「集合意味論」に基づいている．しかしながら，この理論を国際標準リレーショナルデータベース言語 SQL をサポートする商用あるいはオープンソースのリレーショナル DBMS の現場に適用しようとすると，データ操作の対象はタップルの「集合」としての「リレーション」ではなく，一般に重複したタップルの出現を許す「バッグ」（bag）としての「表」（table）なので，これまで集合意味論のもとで提案してきたアプローチをバッグ意味論に拡張する必要がある．本稿では，集合意味論のもと，意図に基づくアプローチを採ることで初めて更新可能となった直積ビューに焦点を当てて，「バッグ直積ビュー」の更新可能性と，更新変換ルールについて検討を加えた．その結果，バッグ直積ビューも意図に基づくアプローチのもとで，更新可能であることが示された．

キーワード ビュー更新問題，ビューサポート，意図に基づくアプローチ，バッグ意味論，集合意味論，SQL，リレーショナルデータベース

1. はじめに

リレーショナルデータベースにビュー(view)を導入したのは，リレーショナルデータモデルを提案したコッド(E. F. Codd)自身である[1]．ビューはリレーショナル代数表現がリレーショナル代数演算を再帰的に適用することで定義され，次のような効用があるとされ，理論的にも実践的にも多大の関心を集め続けてきた：

(i) ユーザが発行する問合せの結果をあたかも実リレーションのように扱える簡略化表現(short-hand)でユーザの利便性に供することができる．

(ii) ANSI/X3/SPARC のいう論理的データ独立性をリレーショナルデータモデルで達成する手段である．

(iii) データベースのセキュリティ実現のための仕掛けとして機能する．

しかしながら，ビューはデータベースに格納されている実リレーションではなく，実リレーション群に対して発行された問合せの結果リレーションを導く「定義」をビューと称している「仮想的」なリレーションにしか過ぎないので，ビューを問合せの対象とするときには問題は生じないが，ビューを「更新」しようとするとビューの更新可能性が問題となる．これを**ビュー更新問題**(view update problem)という．この問題は，理論的にも実用的にもこれまで多くの研究者や実務家の関心を引き，数多くの研究・開発が報告されてきたが，単に構文的あるいは関数的なアプローチ[2]では問題解決が難しく，極度に意味的な問題を孕んでいることが示された[3]．その問題解決のために，インタラクティブなシステムの構築[4]が報告されたほどである．

さて，このような従来のビューサポートの限界を打ち破るべく，「更新意図の外形的推測に基づくアプローチ」（簡単に，意図に基づくアプローチという）が増永により提案された[5, 6]．この理論はビューを一時的に体現化する（materialize）することにより，ビューの更新意図を一意に推測できる場合があるとする理論で，これまでのビューサポートの限界を打破できる可能性を秘めている．

しかしながら，この理論は純粋にリレーショナルデータモデルに立脚して展開されている，つまり「集合意味論」（set semantics）に基づいているので，この理論を国際標準リレーショナルデータベース言語 SQL をサポートする商用あるいはオープンソースのリレーショナル DBMS の現場に適用しようとすると，SQL が操作の対象とするデータは「リレーション」，つまりタップルの「集合」ではなく，「表」（table），つまり一般に重複したタップルの出現を許す「バッグ」（bag），「マルチ集合」（multiset）ともいう，であるので，「集合意味論」のもとで論じてきた「意図に基づくアプローチ」をバッグ意味論に拡張して，「バッグ意味論に基づくビュー更新問題」を検討する必要に迫られることとなった．本稿はこのための第一報であり，集合意味論のもと，意図に基づくアプローチを採ることで初めて更新可能となった直積ビューに焦点を当てて，「バッグ直積ビュー」の更新可能性と，更新変換ルールについて検討を加えている．その結果，バッグ直積ビュー

は集合意味論の場合と同様に，意図に基づくアプローチのもと，更新可能であることが示されている．

以下，本報告では，2 章でバッグ代数を規定し，第 3 章で意図に基づくアプローチの簡単な紹介を行った後，第 4 章でバッグ代数とバッグ直積ビューの更新可能性を意図に基づくアプローチのもとで論じる．第 5 章はまとめと今後の課題を述べる．

2. リレーショナル代数からバッグ代数へ

2.1 リレーショナル代数とビュー

集合意味論に基づくリレーショナルデータモデルのデータ操作言語には大別すると，リレーショナル代数とリレーショナル論理があり，両者は質問記述能力において等価である．リレーショナル代数は次の 8 つの演算から成り立つ:和集合演算(∪),差集合演算(－),共通集合演算（∩），直積演算（×），射影演算（π），選択演算（σ），結合演算（⋈），商演算（÷）．演算の独立性を考えると，和集合演算，差集合演算，直積演算，射影演算，そして選択演算の 5 つでよいことが知られている.リレーショナルデータモデルにおける「ビュー」(view) は実リレーション群にこれらの演算を任意に有限回適用して得られるリレーショナル代数表現そのものと定義される．

2.2 バッグ

バッグ意味論に基づくビュー更新問題を論じるためには，まずリレーショナルデータモデルが前提とする「集合」(set) を拡張して「バッグ」(bag) を定義する必要がある．

【定義 1】(バッグの定義)

リレーションスキーマを $\boldsymbol{R}(A_1, A_2, \ldots, A_n)$とする．$\Omega_{\boldsymbol{R}}=\{A_1, A_2, \ldots, A_n\}$，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．

$$R=\{t_i(k_i) \mid t_i\in dom(\boldsymbol{R}) \ \wedge \ k_i\geqq 1\}$$

ここに，ki は R 中に生起するタップル ti の重複度（multiplicity）を表す．バッグ R の濃度（cardinality），$|R|_{bag}$ と表す，は有限とし，$|R|_{bag}=\Sigma_{i=1,..,p}\ k_i$ と定義する．また，$(\forall t_i(k_i), t_j(k_j)\in R)(t_i=t_j\Rightarrow k_i=k_j)$である．ここに，⇒は含意（logical implication）を表す．□

【定義 2】(バッグの等号)

リレーションスキーマ $\boldsymbol{R}(A_1, A_2, \ldots, A_n)$のバッグ R とバッグ S が等しい，これを $R=_{bag}S$ と書く，とは次が成立しているときをいう．

$$R=bagS\Leftrightarrow(\forall t\in dom(R))(\ t(k)\in R\Leftrightarrow t(k)\in S)$$

ここに，⇔は等価（logical equivalence）を表す．□

バッグをこのように定義することは，古くは Dayal らの論文[7]にその原型を見ることができる．バッグをプログラミング言語の観点や問合せ言語の観点から議論した研究が報告されている[9, 10, 11 他]．ただし，ビューとの関連が論じられているわけではない.また，バッグ理論は集合論の拡張として，数学でも研究対象となっており，例えば，Blizard は G. Cantor が「集合とは異なる元の集まり」と定義したことを紹介したうえで，バッグ，つまりマルチ集合を“A multiset is a collection of elements in which elements are allowed to repeat; it may contain a finite number of indistinguishable copies of a particular element.”と定義している[8]．ここで注意しておくべきことは，indistinguishable copies といっているところである．つまり，コピーは「区別できない」といっているのである．したがって，これをデータベース流に解釈すると，例えば，R={1, 1}と書いてもあまり意味がない．つまり，最初のタップル 1 と 2 番目のタップル 1 とに何か差があるか，換言すれば，R から最初の 1 を削除するというようなことに意味があるかというと，それはないということである．

なぜならば，最初の 1 と 2 番目の 1 は indistinguishable であるからである.したがって，R={1(2)}と書くのが，理にかなっている．定義 1 は，この論旨でそうしているといってよい．

【系 1】リレーションスキーマ $\boldsymbol{R}$ のバッグ $R=\{t_i(k_i) \mid t_i \in \Omega_{\boldsymbol{R}} \ \wedge \ k_i\geqq 1\}$が「リレーション」であるのは，$(\forall i)(k_i=1)$であるとき，及びそのときのみである．

（証明）定義 1 とリレーションの定義より明らか．□

次に,リレーションスキーマ $\boldsymbol{R}$ 上の全てのバッグ(当然，リレーションを含む)がなす集合を定義しておく．

【定義 3】リレーションスキーマ $\boldsymbol{R}$ 上の全てのバッグのなす集合 $\boldsymbol{B_R}$ を次のように定義する．

$\boldsymbol{B_R}=\{R \mid R$ は $\boldsymbol{R}$ のバッグである$\}$ □

2.3 バッグ代数

さて，演算の対象がリレーションからバッグに拡張された状況において，リレーショナル代数の演算子をどのように拡張してバッグ代数を定義するのか，従来示されてきた知見にも触れながらまとめる．

(1) 重複タップルの除去演算子：δ

重複タップルを除去する演算子 δ を次のように定義する．

$$\delta : \boldsymbol{B_R} \rightarrow \boldsymbol{B_R}$$

ここに，$R=\{t_1(k_1), t_2(k_2), \ldots, t_p(k_p)\}$をバッグとするとき，R に δ を施した結果は次の通りである：

$\delta(R)=\{t_1, t_2, \ldots, t_p\}$，ここに $t_1(1)$は単に t_1 と書いた．

$R=\{t_i(k_i) \mid i=1, 2, \ldots, p\}$とするとき，$|R|_{bag}=\Sigma_{i=1,..,p}\ k_i$ と定義したが，|R|=p，つまり|R|は R の異なるタップルの数を表すとする．

【定義 4】(多重度関数)

R をリレーションスキーマ $\boldsymbol{R}(A_1, A_2, \ldots, A_n)$のバッグ

とし，t∈δ(R)とするとき，多重度関数 m を次のように定義する．

$$m(R, t)=k \Leftrightarrow t(k)\in R$$

(2)　バッグ加法和演算子：$\cup_{bag}$

バッグの加法和（additive union）をとる演算子を$\cup_{bag}$と書くことにする．R と S を和両立，すなわち同じリレーションスキーマの異なるインスタンスとしてのバッグ同士としたときに，その加法和は次のように定義される．

$$\cup_{bag}:\ \boldsymbol{B_R}\times\boldsymbol{B_R}\rightarrow\boldsymbol{B_R}$$

ここに，$R=\{t_1(k_1), t_2(k_2), ..., t_p(k_p)\}$と $S=\{u_1(\ell_1), u_2(\ell_2), ..., u_q(\ell_q)\}$を和両立なバッグとするとき，R と S の加法和は次のように定義される．

$R\cup_{bag}S=\{t_i(k_i) \mid t_i\in\delta(R)-\delta(S)\}\cup\{t_i(k_i+\ell_j) \mid t_i=u_j\in\delta(R)\cap\delta(S)\}\ \cup\{u_j(\ell_j) \mid u_j\in\delta(S)-\delta(R)\}$

ここに，$t_i=u_j\in\delta(R)\cap\delta(S)$は $t_i\in\delta(R)\wedge u_j\in\delta(S)\wedge t_i=u_j$ の短縮形である．

なお，「バッグ和」については 2 つの定義法がある．上記の「加法和」は SQL の UNION ALL に相当する．しかし，「最大和」（maximum union）という定義がある．これはδ(R)∩δ(S)のタップルに対して，例えば$t_i=u_j(=t)\in\delta(R)\cap\delta(S)$である場合，最大和に現れる t の重複度を $\max(k_i, \ell_j)$と定義するものである．この定義は，以下に示すバッグ共通演算でタップルの多重度が $\min(k_i, \ell_j)$と定義されることと対照的であることを特長としている．それらの差異を Dayal らは論じているが[7]，本報告の目的は従来のビュー更新可能性の議論を SQL の世界に拡張することであるから，最大和ではなく加法和をとる．

(3)　バッグ差演算子：$-_{bag}$

バッグ差をとる演算子を$-_{bag}$と書くことにする．

$$-_{bag}:\ \boldsymbol{B_R}\times\boldsymbol{B_R}\rightarrow\boldsymbol{B_R}$$

ここに，$R=\{t_1(k_1), t_2(k_2), ..., t_p(k_p)\}$と $S=\{u_1(\ell_1), u_2(\ell_2), ..., u_q(\ell_q)\}$を和両立なバッグとするとき，R と S のバッグ差は次のように定義される．

$R-_{bag}S=\{t_i(k_i) \mid t_i\in\delta(R)-\delta(S)\}\cup\{t_i(\dot{-}(k_i, \ell_j)) \mid t_i=u_j\in\delta(R)\cap\delta(S)\}$

ここに，$\dot{-}(k_i, \ell_j)$は次のように定義される．

$$\dot{-}(k_i, \ell_j)=\begin{cases} k_i-\ell_j, & k_i>\ell_j \\ 0, & \text{そうでない場合}\end{cases}$$

なお，一般に，$t(0)\in R$はタップル t がバッグ R に存在しないことを表す．

【命題 1】リレーションスキーマ $\boldsymbol{R}(A_1, A_2, ..., A_n)$のバッグ R とバッグ S が等しいとき（$R=_{bag}S$），$R-_{bag}S=\phi$(空集合)である．

（証明）定義 2 とバッグ差演算子の定義からほぼ明らか．□

(4)　バッグ共通演算子：$\cap_{bag}$

バッグ共通をとる演算子を$\cap_{bag}$と書くことにする．

$$\cap_{bag}:\ \boldsymbol{B_R}\times\boldsymbol{B_R}\rightarrow\boldsymbol{B_R}$$

ここに，$R=\{t_1(k_1), t_2(k_2), ..., t_p(k_p)\}$と $S=\{u_1(\ell_1), u_2(\ell_2), ..., u_q(\ell_q)\}$を和両立なバッグとするとき，R と S のバッグ共通は次のように計算される．

$R\cap_{bag}S=\{t_i(\min(k_i, \ell_j)) \mid t_i=u_j\in\delta(R)\cap\delta(S)\}$

ここに，$\min(k_i, \ell_j)$は k_i と ℓ_j の小さな値の方をとる演算である．

(5)　バッグ直積演算子：$\times_{bag}$

R と S をリレーションスキーマ $\boldsymbol{R}(A_1, A_2, ..., A_n)$と $\boldsymbol{S}(B_1, B_2, ..., B_m)$のバッグとする．このとき，R と S の直積演算子を$\times_{bag}$と書くことにする．

$$\times_{bag}:\ \boldsymbol{B_R}\times\boldsymbol{B_S}\rightarrow\boldsymbol{B_{R\times S}}$$

ここに，$\boldsymbol{R}\times\boldsymbol{S}$はリレーションスキーマ $\boldsymbol{R}(A_1, A_2, ..., A_n)\times\boldsymbol{S}(B_1, B_2, ..., B_m)$を表し，$\boldsymbol{B_{R\times S}}=\{T \mid T$ は $\boldsymbol{R}\times\boldsymbol{S}$ のバッグである$\}$と定義される．

このとき，$R=\{t_1(k_1), t_2(k_2), ..., t_p(k_p)\}$と $S=\{u_1(\ell_1), u_2(\ell_2), ..., u_q(\ell_q)\}$をバッグとするとき，R と S のバッグ直積は次のように定義される．

$R\times_{bag}S=\{(t_i, u_j)(k_i\times\ell_j) \mid t_i(k_i)\in R)\wedge u_j(\ell_j)\in S\}$

(6)　バッグ射影演算子：π_{bag}

R をリレーションスキーマ $\boldsymbol{R}(A_1, A_2, ..., A_n)$のバッグとする．$X=\{A_{i1}, A_{i2}, ..., A_{ir}\}$（$\subseteq\Omega_{\boldsymbol{R}}$）とするとき，R の X 上のバッグ射影$\pi_{bagX}R$は次のように定義される．

$$\pi_{bagX}:\ \boldsymbol{B_R}\rightarrow\boldsymbol{B_{R[X]}}$$

ここに，$\boldsymbol{R}[X]$はリレーションスキーマ $\boldsymbol{R}(A_1, A_2, ..., A_n)[A_{i1}, A_{i2}, ..., A_{ir}]$を表し，$\boldsymbol{B_{R[X]}}=\{T \mid T$ は $\boldsymbol{R}[X]$のバッグである$\}$と定義される．

このとき，$R=\{t_1(k_1), t_2(k_2), ..., t_p(k_p)\}$とすれば，バッグ R の属性集合 X 上のバッグ射影演算は次のように定義される．

$\pi_{bagX}(R)=\{x(k) \mid (\exists t_i\in\delta(R))(x=\pi_X(t_i))\wedge k=\Sigma_j k_j$ such that $\pi_X(t_j)=x\}$

(7)　バッグ選択演算子：σ_{bag}

バッグ選択をとる演算子をσ_{bag}と書くことにする．

$$\sigma_{bag}:\ \boldsymbol{B_R}\rightarrow\boldsymbol{B_R}$$

ここに，$R=\{t_1(k_1), t_2(k_2), ..., t_p(k_p)\}$をバッグとし，P をδ(R)が満たさないといけない述語(predicate)とする．このとき，R に対して P に関するバッグ選択演算は次のように定義される．

$\sigma_{bagP}(R)=\{t_i(k_i) \mid t_i\in\delta(R)\ \wedge\ P(t_i)=TRUE\}$

(8)　バッグ結合演算子：$\bowtie_{bag}$

R と S をリレーションスキーマ $\boldsymbol{R}(A_1, A_2, ..., A_n)$と $\boldsymbol{S}(B_1, B_2, ..., B_m)$のバッグとする．このとき，R と S の属性 Ai と Bj 上のθ-結合演算の結果を$R\bowtie_{bag\ Ai\,\theta\,Bj}S$と書くことにする．

$$\bowtie_{bag\ Ai\,\theta\,Bj}:\ \boldsymbol{B_R}\times\boldsymbol{B_S}\rightarrow\boldsymbol{B_{R\bowtie Ai\,\theta\,Bj S}}$$

ここに，$\boldsymbol{R}\bowtie_{Ai\,\theta\,Bj}\boldsymbol{S}$はリレーションスキーマ $\boldsymbol{R}(A_1, A_2, ...,$

A_n) $\bowtie_{Ai\,\theta\,Bj}$***S***(B_1, B_2, ..., B_m))を表し，$\boldsymbol{B}_{\boldsymbol{R}\bowtie Ai\,\theta\,Bj\boldsymbol{S}}$={T | T は $\boldsymbol{R}\bowtie_{Ai\,\theta\,Bj}\boldsymbol{S}$ のバッグである}と定義される．

このとき，R={$t_1(k_1)$, $t_2(k_2)$, ..., $t_p(k_p)$}と S ={$u_1(\ell_1)$, $u_2(\ell_2)$, ..., $u_q(\ell_q)$}を，$\Omega_R \cap \Omega_S \neq \phi$ (空集合)であるバッグとするとき，R と S のバッグ結合演算は次のように定義される．

$R\bowtie_{bag\ Ai\,\theta\,Bj}S = \{ (t_i, u_j)(k_i \times \ell_j) \mid t_i \in \delta(R) \wedge u_j \in \delta(S) \wedge \pi_{\Omega R \cap \Omega S}(t_i) = \pi_{\Omega R \cap \Omega S}(u_j)\}$

バッグ同士の自然結合演算を明示的に考察の対象にする場合，バッグ自然結合演算子を$*_{bag}$ と書くことにする．

$$*_{bag}:\ \boldsymbol{B_R} \times \boldsymbol{B_S} \rightarrow \boldsymbol{B_{R*S}}$$

ここに，R={$t_1(k_1)$, $t_2(k_2)$, ..., $t_p(k_p)$}と S ={$u_1(\ell_1)$, $u_2(\ell_2)$, ..., $u_q(\ell_q)$}を，$\Omega_R \cap \Omega_S \neq \phi$ (空集合)であるバッグとするとき，R と S のバッグ自然結合は次のように計算される．

$R *_{bag} S = \{ (t_i * u_j)(k_i \times \ell_j) \mid t_i \in \delta(R) \wedge u_j \in \delta(S) \wedge \pi_{\Omega R \cap \Omega S}(t_i) = \pi_{\Omega R \cap \Omega S}(u_j)\}$

3. バッグビューとビュー更新問題

3.1 バッグビューの定義

バッグ意味論に基づくビュー，これを簡単に「バッグビュー」という,の定義を与えることから始めるが，実リレーションあるいは実表とビューの置かれている立場を峻別すると，次の 3 つの分類が可能であることに注意する．

(a) 実リレーションあるいは実表，そしてビュー共に集合意味論に基づく．
(b) 実リレーションあるいは実表は集合意味論に基づくが，ビューはバッグ意味論に基づく．
(c) 実リレーションあるいは実表，そしてビュー共にバッグ意味論に基づく．

(a)が従来，ビュー更新問題が論じられてきた枠組みである．意図に基づくアプローチもこの範疇に入る．(b)は従来のリレーショナル代数で，射影演算によりタップルの重複が発生する状況を反映する状況を論じるのに適した枠組みである.(c)は実際に SQL ではバッグを実表として生成することも可能な状況を念頭に置いており，本稿はこの枠組みでビュー更新を論じる．(a)を集合 - 集合意味論，(b)を集合 - バッグ意味論，(c)をバッグ - バッグ意味論といったりする（本稿ではこの場合を単純に「バッグ意味論」と言っている).

【定義 5】(バッグ意味論でのビュー定義)

1．実表 R はビューである．
2．V1，V2 をビューとするとき，V1 と V2 が和両立ならば，$V1 \cup_{bag} V2$，$V1 -_{bag} V2$，$V1 \cap_{bag} V2$ もビューである．
3．V1，V2 をビューとするとき，$V1 \times_{bag} V2$ はビューである．
4．V をビューとするとき，$\pi_{bag X}(V)$はビューである．ここに，X は V の属性集合である．
5．V をビューとするとき，$\sigma_{bag Ai\,\theta\,Aj}(V)$はビューである．ここに，Ai と Aj は θ -比較可能とする．
6．V1，V2 をビューとするとき，$V1 \bowtie_{bag Ai\,\theta\,Bj} V2$ はビューである．ここに，Ai と Bj は θ -比較可能とする．
7．1．から 6．で定義されるもののみが，ビューである．

なお，自然結合ビュー$V1 *_{bag} V2$ は陽には定義されていないが，上述の定義から容易に導ける．

3.2 ビュー更新可能性の定義

リレーショナルデータベースのビュー更新問題に対して,当初から取り組まれてきた構文的(syntactic)アプローチ[2]をまず述べる：ビューは実リレーション群がなすデータベース状態(database state)からビュー状態(view state)への「関数」(function)であると規定することから始まる．つまり，s_τをある時刻 τ におけるデータベースの状態，V をビュー定義，$V(s_\tau)$は（その時刻 τ における)ビューの状態，$\boldsymbol{u}$ を $V(s_\tau)$に対して発行された更新操作とするとき，$\boldsymbol{u}$ 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副作用がなく (no side effects) かつ一意 (unique) な変換 (translation) T が存在するときと定義する．すなわち，図 1 の可換図式(commutative diagram)が成立するときをいう．なお，変換 T に副作用がないとは，$\boldsymbol{u}(V(s_\tau)) = V(T(\boldsymbol{u})(s_\tau))$ が成立することを意味する．変換 T に一意性を課したことは議論の余地があるところだとしながらも，その理由は 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としている．

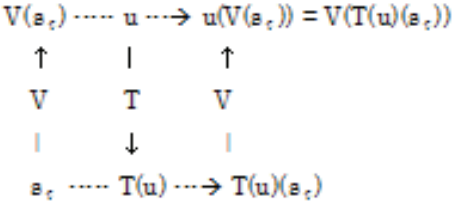

図 1　時刻 τ においてビュー更新要求 $\boldsymbol{u}$ が変換可能であることを示す可換図式

【定義 6】(ビューの更新可能性)

V をビューとする．V が更新可能とは，いかなる時刻 τ，いかなる更新要求 $\boldsymbol{u}$ に対しても，図 1 の可換図式が成立するときをいう．ここに，更新とは削除，挿入，あるいは書換のいずれかとする．

ここで，注意しておきたいことは，上記は本来集合意味論のもとで与えられた定義であるが，この定義はバッグ意味論のもとでも変わらない，つまり実表がバ

ッグでもビューがバッグでも，ビューが更新可能とは図 1 に示した「可換図式」が成立するとき，およびそのときのみをいうということに変わりはないということである．その理由はほとんど明らかであろう．

3.3 集合意味論でのビューの更新可能性

どのようなビューがどのようなときに，更新可能となるのであろうか．従来，集合意味論のもとで示されてきた結果を表示する[5]．7 つの基本ビューに対する意図に基づくアプローチによる更新可能性が表 1 第 3 列に，従来型アプローチによる結果が同表第 4 列に示されている．

表 1　7 つの基本ビューの更新可能性の比較一覧

基本ビューの種類	更新操作	意図に基づくアプローチによる更新可能性	従来型アプローチによる更新可能性
和集合ビュー	削除	○	○
	挿入	×*	×
	書換	○(直接書換法)	×
差集合ビュー	削除	×*	×
	挿入	○	○
	書換	×*	×
共通集合ビュー	削除	×*	×
	挿入	○	○
	書換	×*	×
直積ビュー	削除	◎	×
	挿入	◎	×
	書換	◎(書換要求が両立)	×
射影ビュー	削除	○	○
	挿入	○(主キーを含む)	○(主キーを含む)
	書換	○(直接書換法)	×
選択ビュー	削除	○	○
	挿入	○	○
	書換	○	○
結合ビュー	削除	◎	×
	挿入	◎	×
	書換	◎(書換要求が両立)	×

○，×：従来型アプローチで更新可能，不可能
◎：意図に基づくアプローチで更新可能
括弧書きは条件，＊は本質的に不可能，を表す．

4. バッグビューの更新可能性

4.1 バッグ意味論でのビューの更新可能性

さて，集合意味論に基づくリレーショナル代数で定義されたビューのうち，基本的な 7 つのビューの更新可能性は表 1 にその結果を示した通りである．然らば，バッグ代数のもとで定義されるビュー，これを「バッグビュー」という，の更新可能性はどのように規定されるのであろうか，みていく．

まず，集合意味論はバッグ意味論の特殊な場合であるので，バッグ意味論のもとで可能なビュー更新は集合意味論のもとでも可能である．しかし，一般に集合意味論のもとで可能とされたビュー更新がバッグ意味論のもとでも可能とはならないだろう．反面，バッグ意味論のもとでは不可能とされたビュー更新が集合意味論のもとでは可能となるかもしれない．

さて，上述の「バッグ意味論のもとで可能なビュー更新は集合意味論のもとでも可能である」の対偶をとれば,「集合意味論のもとで不可能なビュー更新はバッグ意味論のもとでも不可能である」ので，表 1 の第 3 列で×印がついているビューについては，バッグ意味論でも×となる．したがって，バッグ意味論でのビュー更新可能性は表 1 第 3 列で，○，あるいは◎である場合について，バッグビューの更新可能性を検証することが必要であり重要であることが分かる．

なお，今回は紙面の都合上，バッグ直積ビューの更新可能性を意図に基づくアプローチのもとで検証する．直積ビューを検証の最初のターゲットにしたのは，それが従来型アプローチでは更新不可であったが，集合意味論のもと意図に基づくアプローチを採用することで初めて更新可能となったからである．

4.2 バッグに対する更新とその意味

バッグ意味論でのビュー更新問題を論じる前提として，一般にバッグに対するタップルの更新を論じておく必要がある．

(1)　バッグへの（重複）タップル群の挿入

バッグ R に対して，（重複）タップル（群）の挿入要求 $\boldsymbol{i}$ があったとする．

$\boldsymbol{i}$ = insert into R values S;

ここに，S は R と和両立とし，$S = \{t_i(k_i) \mid t_i \in \Omega_R \wedge k_i \geqq 1\}$とする．

$\boldsymbol{i}$ は常に受け入れられて，その結果は次のように定義される．

$$\boldsymbol{i}(R) = R \cup_{\mathbf{bag}} S$$

また，$\boldsymbol{i}$ の意味（meaning）であるが，$t_i=(a_{i1}, a_{i2}, \ldots, a_{in}) \in \delta(S)$とするとき，$(\forall i)((R.A_1= a_{i1}) \wedge (R.A_2= a_{i2}) \wedge \ldots \wedge (R.A_n= a_{in}))$が成立することをいう．

(2)　バッグからのタップル削除

バッグ R から，dom($\boldsymbol{R}$)上の述語 P を満たすタップル群を削除する要求を $\boldsymbol{d}$ とする．

$\boldsymbol{d}$ = delete from R where P;

したがって，$S=\{t_i(k_i) \in R) \mid t_i \in \delta(R) \wedge P(t_i)=\mathrm{TRUE}\}$とすれば，削除の結果は次のように定義される．

$$\boldsymbol{d}(R) = R -_{\mathrm{bag}} S$$

つまり，削除は，バッグに探索条件を与えて，その探索条件に合致するタップル群を全てバッグから削除する．バッグには同じ探索条件を満たすタップルは，その多重度分だけ存在するから，そのような重複したタップル群は探索条件を満たせば，バッサリ削除されることになる．つまり，探索条件を満たすタップル群

の一部を削除するというような操作はない．これは，重複したタップル群はお互いに indistinguishable であるから，そのうちのどれを削除するというような指定はできないと考えるのが理に適っていること（このことは，2.2 節で述べた）と符合する．

また，***d*** の意味であるが，$t_i=(a_{i1}, a_{i2}, \ldots, a_{in}) \in \delta(S)$ とするとき，$(\forall i) \neg ((R.A_1= a_{i1}) \wedge (R.A_2= a_{i2}) \wedge \ldots \wedge (R.A_n= a_{in}))$が成立することをいう．

（3） バッグのタップル書換

バッグのタップル群の書換は，探索条件を満たすタップルの指定された属性の値を「値式」に基づいて書き換える操作である．削除の場合と同様に，重複したタップルは全てを同じように書き換えることで，書換を行う．重複したタップル群の一部だけを書き換えることはしないということである．

バッグ R の，Ω_R 上の述語 P を満たすタップル群の属性値を値式に基づき書き換える要求を ***r*** とする．

r = update R set $A_{i1}=vexp_{i1}, \ldots, A_{iq}=vexp_{iq}$ where P;

つまり，削除の時と同じく述語 P により更新の対象となる R のタップル群が同定され，それらの属性値を指定された値式$(vexp_{i1}, \ldots, vexp_{iq})$に基づき更新する．

r を実現するには 2 つの方法がある．

理論的アプローチ：R からまず述語 P を満たすタップルの集合を削除する．続いて，削除されるべきタップル集合の各タップルを更新した結果を挿入する．従って，述語 P を満たすタップルの集合の削除が不可能ならば，あるいは更新結果の挿入が不可能であれば，このタップル書換要求は受け付けられない．これは，従来のリレーショナル代数での書換の考え方に同じである．

直接書換法：R からまず述語 P を満たすタップルの集合を同定し，それを削除するのではなく，その集合の各タップルの属性値を値式に基づき更新していく．

直接書換法により，集合意味論での和集合ビューや射影ビューの書換が可能となる[5].

4.3 バッグ直積ビューの更新可能性

集合意味論のもと，意図に基づくアプローチを採ることで初めて更新可能となった直積ビューに焦点を当てて，「バッグ（意味論のもとでの）直積ビュー」の更新可能性と，更新変換ルールについて検討を加える．

4.3.1 バッグ直積ビューの削除可能性の検討

意図に基づくアプローチのもと，バッグ直積ビューの削除が許される場合，許されない場合があることを例示する．

【例題 1】R(A)={1(2), 2}，S(B)={1, 2(2)}を実表とし，バッグ直積ビューV を次のように定義する．

$V = R \times_{bag} S = \{(1, 1)(2), (1, 2)(4), (2, 1), (2, 2)(2)\}$

① V に対して次の削除要求が発せられたとする．

d1=delete from V where (A=2 AND B=1) OR (A=2 AND B=2);

つまり，V から{(2, 1), (2, 2)(2)}を削除したいという要求である．

まず，従来型アプローチでは直積ビューへの更新要求は受け付けない．一方，意図に基づくアプローチでは更新要求が受付られる可能性があるので，この削除要求がどのように処理されるか見てみる．

削除要求は where 句に記されている通り，(R.A=2 AND S.B=1) OR (R.A=2 AND S.B=2)である．換言すると，V のタップルのうち A 値が 2 で B 値が 1 のタップル，あるいは A 値が 2 で B 値が 2 のタップルは V にあることの意味を失ったので，削除したいとの要求が出たということである．つまり，削除の意味から，¬((R.A=2 AND S.B=1) OR (R.A=2 AND S.B=2))=TRUE ということで，これは即ち，((¬(R.A=2) OR ¬(S.B=1)) AND (¬(R.A=2) OR ¬(S.B=2)))=TRUE ということである．左辺は連言標準形になっているので，これを分配則を用いて選言標準形に変換すると，次のようになる．

((¬(R.A=2) AND ¬(R.A=2)) OR (¬(R.A=2) AND ¬(S.B=2)) OR (¬(S.B=1) AND ¬(R.A=2)) OR (¬(S.B=1) AND ¬(S.B=2)))=TRUE.

つまり，この意味処理の言っていることは，所望の削除要求 ***d1*** は次の 4 つの代替案のどれかで実現できるかもしれないと示唆したということである（ただし，いずれかの代替案が副作用のない削除を実現するかどうかは保証の限りではない）．

(a) delete from R where A=2;
(b) delete from R where A=2;
　　delete from S where B=2;
(c) delete from R where A=2;
　　delete from S where B=1;
(d) delete from S where B=1 AND B=2;

そこで，意図に基づくアプローチにより，仮に V をマテリアライズし ***d1*** を適用して ***d1***(V)を得て，一方で，これら 4 つの代替案をそれぞれ実行して，その結果得られるビューを改めてマテリアライズしてみて，もしそれらのうちどれか一つが唯一に ***d1***(V)に等しくなれば，それが削除要求の動機を反映していると推測できるので，その代替案でビュー更新を行うこととする．

この例では，唯一に代替案(a)のみがそれを満たすことが検証できるので，意図に基づくアプローチのもと，バッグ直積ビューへの削除要求は受理される場合のあることが示された．

勿論，受理されない V への削除要求もある．

② V に対して次の削除要求が発せられたとする．

d2=delete from V where A=2 AND B=2;

この場合，削除の意味から，(￢(R.A=2) OR ￢(S.B=1))=TRUE ということであるから，この削除を実現できるとすれば，次の 2 つの代替案のいずれかに依らねばならない．

(a) delete from R where A=2;

(b) delete from S where B=1;

しかしながら，容易に検証できるように，いずれの代替案でも副作用が起こり，所望の削除は実現できないことが，ビューを一時的にマテリアライズしてみるとわかるので，この削除要求は受け付けられない．

従って，次の命題を示すことができた．

【命題 2】意図に基づくアプローチのもとで，バッグ直積ビューは削除可能である．

（証明）例題 1 の議論より明らか．□

4.3.2 バッグ直積ビューの挿入可能性の検討

意図に基づくアプローチのもと，バッグ直積ビューの挿入が許される場合，許されない場合があることを例示する．

【例題 2】R(A)={1(2), 2}，S(B)={1, 2(2)}を実表とし，バッグ直積ビューV を次のように定義する．

$V = R \times_{bag} S = \{(1, 1)(2), (1, 2)(4), (2, 1), (2, 2)(2)\}$

① V に対して次の挿入要求が発せられたとする．

i1= insert into V values (3, 1), (3, 2)(2);

まず，従来型アプローチでは直積ビューへの挿入要求は受け付けられない．

次に，意図に基づくアプローチでこの挿入要求がどのように処理されるか見てみる．

挿入要求は values 句に記されていることから分かるように，***i1***(V)∋(3, 1) AND ***i1***(V)∋(3, 2)(2)である．換言すると，(3,1), (3,2), (3,2)が新たに V のタップルとしての市民権を得たと主張していることになる．そうすると，バッグ意味論なので，まず R に対して次の 4 つの代替案を考えないといけないことが分かる．

(a) R に何も挿入しない

(b) insert into R values (3);

(c) insert into R values (3)(2);

(d) insert into R values (3)(3);

一方，S に対しては次の 5 つの代替案を考えないといけないことが分かる．

(e) S に何も挿入しない

(f) insert into S values (1);

(g) insert into S values (2);

(h) insert into S values (1), (2);

(i) insert into S values (1), (2)(2);

(a)-(e)の組合せでは ***i1*** を実現しようがないので，その他の組合せ，4×5－1=19 個の組合せについて，ビューをマテリアライズして，意図に基づくアプローチのもと実現可能性を検証することとなるが，発見的手法として，この時点では R にはタップル(3)がないので(a)との組み合わせは検証しなくてよい．従って，19－4=15 個の組合せについて，検証する．その結果，唯一，(b)-(e)の組合せだけが副作用を引き起こすことなく，丁度 ***i1*** を実現することが分かるので，意図に基づくアプローチのもと，バッグ直積ビューへの挿入要求は受理される場合のあることが示される．

勿論，受理されない V への挿入要求もある．

② V に対して次の挿入要求が発せられたとする．

i2= insert into V values (1, 1)(2);

まず，従来型アプローチでは直積ビューへの挿入要求は受け付けられない．

次に，意図に基づくアプローチでこの挿入要求がどのように処理されるか見てみる．

挿入要求は values 句に記されている通り，***i2***(V)∋(1, 1)(2)である．換言すると，(1,1), (1,1)が新たに V のタップルとしての市民権を得たと主張していることになる．そうすると，バッグ意味論なので，まず R に対して次の 3 つの代替案があることが分かる．

(a) R に何も挿入しない

(b) insert into R values (1);

(c) insert into R values (1)(2);

一方，S に対しては次の 3 つの代替案があることが分かる．

(d) S に何も挿入しない

(e) insert into S values (1);

(f) insert into S values (1)(2);

(a)-(d)の組合せでは ***i2*** を実現しようがないので，その他の 8 個の組合せについて，ビューをマテリアライズして，意図に基づくアプローチのもと実現可能性を検証することとなる．(a)-(e)では副作用が起こることが分かり，従って，(a)-(f)を試す必要はない．(b)-(d)では ***i2*** を実現できないことは容易に検証され，(b)-(e)では副作用が発生し，従って，(b)-(f)を試す必要はない．(c)-(d)では副作用が発生するから，(c)-(e)と(c)-(f)を試す必要はない．つまり，***i2*** を実現する更新変換は存在しないことが分かる．

【命題 3】意図に基づくアプローチのもとで，バッグ直積ビューは挿入可能である．

（証明）例題 2 の議論より明らか．□

4.3.3 バッグ直積ビューの書換挿入可能性の検討

意図に基づくアプローチのもと，バッグ直積ビューの書換が許される場合，許されない場合があることを例示する．

【例題 3】R(A)={1(2), 2}，S(B)={1, 2(2)}を実表とし，バッグ直積ビューV を次のように定義する．

$V = R \times_{bag} S = \{(1, 1)(2), (1, 2)(4), (2, 1), (2, 2)(2)\}$

① V に対して次の書換要求が発せられたとする．

r1= update V set A=3 where (A=2 AND B=1) OR (A=2 AND B=2);

つまり，V 中の{(2, 1), (2, 2)(2)}というタップル群を{(3, 1), (3, 2)(2)}に書き換えたいという要求である．このとき，V から{(2, 1), (2, 2)(2)}を削除したいという要求 ***d1***=delete from V where (A=2 AND B=1) OR (A=2 AND B=2);は例題 1 で検証したように削除可能であった．そして，その削除は，T(***d1***)=delete from R where A=2;で実現されることが示されている．

ところで，***r1*** は R から削除するべき A 値が 2 のタップルの値を 3 に書き換えて欲しいという要求であって，これを実現するには次のタップル挿入要求 ***i3*** が V に対して発行されればよい．

i3=insert into V－{(2, 1), (2, 2)(2)} values (3, 1), (3, 2)(2);

i3 は T(***i3***)=insert into R values (3);で実現できる．このとき，T(***d1***)と T(***i3***)は書換要求 ***r1*** と「両立している」（compatible）という．つまり，意図に基づくアプローチのもと，バッグ直積ビューへの書換要求は受理される場合のあることが示された．

勿論，受理されない V への書換要求もある．

② V に対して次の書換要求が発せられたとする．

r2= update V set B=3 where (A=2 AND B=1) OR (A=2 AND B=2);

where 句の条件を満たすタップル群の削除は，上記①の場合と同じく R への削除要求 T(***d1***)で実現されるが，書換のために発行される挿入要求は次のようになる．

i4=insert into V－{(2, 1), (2, 2)(2)} values (2, 3), (2, 3)(2);

しかし，***i4*** を実現できる実表への変換は存在しないので，この場合の書換要求は受理されない．

【命題 4】意図に基づくアプローチのもとで，バッグ直積ビューは書換可能である．

（証明）例題 3 の議論より明らか．□

以上のバッグ直積ビューに関する更新可能性の議論を纏めると表 2 のようになる．更新変換ルールは各例題で議論したスキームに則る．

表 2　直積ビューに関する更新可能性

	バッグ意味論	集合意味論
削除	◎	◎
挿入	◎	◎
書換	◎（書換要求が両立）	◎（書換要求が両立）

◎　意図に基づくアプローチで更新可能

5. おわりに

SQL はバッグ意味論に基づくデータ操作言語なので，ビュー更新メカニズムを商用あるいはオープンソースのリレーショナル DBMS で実装するにあたっては，従来の集合意味論のもとで考察されてきたビュー更新問題をバッグ意味論のもとで再考察する必要がある．本稿では，集合意味論のもと，これまで最もビューサポート能力が高いと考えられる「更新意図の外形的推測に基づくアプローチ」，即ち，意図に基づくアプローチを採ることで初めて更新可能となった直積ビューに焦点を当てて，「バッグ直積ビュー」の更新可能性と，更新変換ルールについて検討を加えた．その結果，バッグ直積ビューも意図に基づくアプローチのもと，集合意味論の場合と同様に更新可能であることが示された．

今後，バッグ直積ビュー以外のバッグビューの更新可能性に検討を加えると共に，議論したバッグビューの更新変換メカニズムをオープンソースのリレーショナル DBMS である PostgreSQL に実装し，性能を検討しつつ必要な改良を施して，実用に供していく予定である．

【謝辞】本研究は JSPS 科研費 JP16K00152 の助成を受けたものである．

参　考　文　献

[1] E. F. Codd, “Recent Investigations in a Relational Database System”, Information Processing 74, pp. 1017-1021, North-Holland, 1974.

[2] U. Dayal and P. Bernstein, “On the Updatability of Relational Views”, Proc. 4th VLDB, pp.368-377, 1978.

[3] Y. Masunaga, “A Relational Database View Update Translation Mechanism”, Proc. 10th VLDB, pp.309-320, 1984.

[4] A. Sheth, J. Larson and E. Watkins, “TAILOR, A Tool for Updating Views”, LNCS, Vol. 303, pp.190-213, Springer, 1988.

[5] 増永良文，“更新意図の外形的推測に基づくリレーショナルデータベースビューの更新可能性”, 8p., WebDB Forum 2015 会議録，2015 年 11 月．

[6] Y. Masunaga, “An Intention-based Approach to the Updatability of Views in Relational Databases”, Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication (ACM IMCOM 2017), P5-8, Beppu, Japan, 2017.

[7] U. Dayal, N. Goodman and R. H. Katz, “An Extended Relational Algebra with Control over Duplicate Elimination, Proc. PODS, pp.117-123, 1982.

[8] W. D. Blizard, “Multiset Theory”, Notre Dame Journal of Formal Logic, Vol.30, No.1, pp.36-66, 1989.

[9] J. Albert, “Algebraic Properties of Bag Data Types”, Proc. 17th VLDB, pp.211-219, 1991.

[10] S. Grumbach and T. Milo, “Towards Tractable Algebras for Bags”, J. of Computer and System Sciences, 52, pp.570-588, 1996.

[11] L. Libkin and L. Wong, “Query Languages for Bags and Aggregate Functions”, J. of Computer and System Sciences, pp.241-272, 1997.