意図に基づくアプローチのもとでのSQLビューの更新可能性

― ビュー更新判定問題の定式化と実問題への適用に向けた計算量削減の試み ―

長田　悠吾† 　増永　良文†† 　石井　達夫†

† SRA OSS, Inc. 日本支社　〒 464–8603 東京都豊島区南池袋 2–32–8
†† お茶の水女子大学（名誉教授）　〒 112–8610 東京都文京区大塚 2–1–1
E-mail: †{nagata,ishii}@sraoss.co.jp, ††yoshi.masunaga@gmail.com

あらまし　リレーショナルデータベースにおけるビューは，データ独立性、問合せの簡略化、セキュリティの手段を提供する利便性の高い機能である．しかしながら，ビューはこれを定義する問合せが実体であり，実データを持たない．そのため，いかなる条件下でビューが更新可能であるのか，ビューに対する更新をいかに実テーブルの更新に変換するのかが問題となる．このビュー更新問題に対し，我々は「意図に基づくアプローチ」を提案し，従来は不可能とされてきたSQLの直積ビューや結合ビューが更新可能となることを示すと共に，本アプローチをオープンソースのRDBMSであるPostgreSQL上に実装し検証を行ってきた．しかしながら，実装された更新可能性判定アルゴリズムは，考えられる変換候補を総当たりに調べ、その全ての候補についてビューのマテリアライズを実行するというもので，その計算量の大きさが問題となっている．この問題の見通しのよい議論を可能とするため，著者らにより，SQLビューの更新可能性判定が「非線形連立方程式」を解く問題に帰着できることが示されている．本稿ではこのアイデアに基づくビュー更新判定問題の定式化と，更新可能性判定をより効率化する試みについて述べる．

キーワード　ビュー，ビュー更新問題，意図に基づくアプローチ，バッグ意味論，SQL，PostgreSQL，リレーショナルデータベース，非線形連立方程式

1. はじめに

リレーショナルデータベースにおけるビュー (view) はリレーショナルデータモデルの発案者であるコッドにより導入された概念であり [1]，問合せの結果をあたかも実リレーションのように扱う簡略化表現 (short-hand) 手段として，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として，またデータベースのセキュリティ実現のための手段としてユーザに高い利便性を提供している．

しかしながら，ビューは実リレーション群に対する問合せの「定義」が実体であり，その実データがデータベースに格納されているわけではない．そのため，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，物理的には存在していない仮想的なリレーションであるビューを「更新」しようとすると，ビューの更新可能性、および、ビューに対する更新をいかに実テーブルの更新に変換するのかが問題となる．これをビュー更新問題 (view update problem) という．

ビュー更新問題は難解な問題であることが知られており，これまで多くの研究が報告されて来た．従来型のアプローチとして，構文的 [2]，意味的 [3]，そして対話的 [4] アプローチがとられてきたが，いずれも更新可能とされるビューへの制約が強く，ビュー更新問題は完全には解決されていない．これに対し，近年著者らにより「意図に基づくアプローチ」が提案され，従来は不可能とされてきたSQLの直積ビューや結合ビューが更新可能となることが示された．なお，このアプローチは当初，リレーションをタップルの重複のない集合とみなす「集合意味論」を元に理論展開されていたが，実際のリレーショナルデータベースで使用される国際標準リレーショナルデータベース言語SQLでは，表 (table) に重複した行の生起を許す「バッグ意味論」(マルチ集合意味論ともいう）に基づいているため，集合意味論からバック意味論への理論の拡張が行われている [5]．さらに，本アプローチをオープンソースのRDBMSであるPostgreSQL上にプロトタイプを実装することで検証が行われ，実際に期待した通りに，ビューの更新可能性を拡張可能であることが確認されている [6], [7], [8].

しかしながら，このプロトタイピングで実装された更新可能性判定アルゴリズムは，考えられるビューの更新方法の候補を総当たりに調べ、その全ての候補についてビューのマテリアライズ，すなわちビューに対する問合せ結果を実データに変換する操作を行うというもので，その計算量の大きさが問題となっている．この問題の見通しのよい議論を可能とするため，著者らにより，SQL ビューの更新可能性判定が「非線形連立方程式」を解く問題に帰着できることが示されている [11]．本稿ではこのアイデアに基づくビュー更新判定問題の定式化と，更新可能性判定をより効率化する試みについて述べる．

2. ビュー更新問題

ビュー更新問題を説明するため，まずビューの更新可能性を以下に定義する．

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) とみなすことができる．s_τ をある時刻 τ におけるデータベース

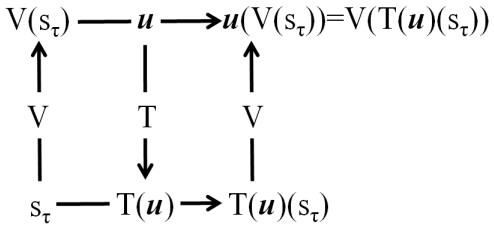

図 1　時刻 τ においてビュー更新要求 u が変換可能であることを示す可換図式

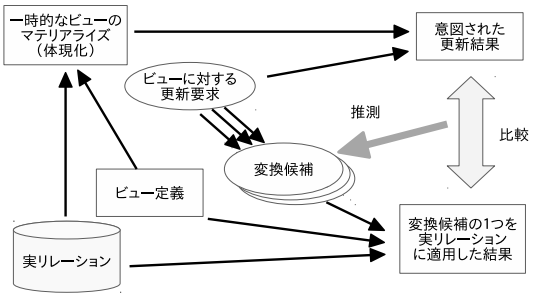

図 2　更新意図の外形的推測に基づくビュー更新可能性

の状態，V をビュー定義，$V(s_\tau)$ を（その時刻 τ における）ビューの状態，u を $V(s_\tau)$ に対して発行された更新操作とする．ここで、ビュー状態 $V(s_\tau)$ への更新要求 u からデータベース状態 s_τ への更新操作 $T(u)(s_\tau)$ への変換 (translation)T を考える．このとき、副作用がない (no side effects) 変換 T が一意 (unique) に存在するとき，すなわち，図 1 の可換図式 (commutative diagra) が成立するときに，u は変換可能 (translatable) であると定義する [2]．ここで，変換 T に副作用がないとは，$u(V(s_\tau)) = V(T(u)(s_\tau))$ が成立することを意味する．

以上の定義のもと，ビュー更新問題とは u を変換可能な変換 T が存在するか、存在する場合はそれがどのようなものであるかを問う問題である．副作用のない変換 T が複数存在し一意に定まらない場合には，曖昧性 (ambiguity) を解消できないとして，そのビューは更新できないとみなす．

3. 意図に基づくアプローチ

ここでは，著者らの提案する意図に基づくアプローチについて述べる（図 2）．

このアプローチは，図 1 の可換図式を満たす変換を「ユーザのビュー更新意図」の推測により特定しようとするものである．具体的には，まずビューを一時的にマテリアライズ (materialize, 体現化) し，ユーザから発行された更新要求を適用することにより，ユーザが望む更新結果，すなわちユーザの意図を得る．本アプローチではこれを「更新意図の外形的推測」と呼んでいる．さらに，複数存在する変換の候補 (alternatives) について，その結果を逐一計算し，上で求めたユーザの望む更新結果と比較し，副作用なくユーザの意図を実現できるかを確認する．全ての変換候補を確認し，ユーザの意図を実現可能な変換候補が唯一存在する場合には，その変換候補の下にビューは更新可能であると判断される．もしそのような候補が見つからない，または複数見つかるような場合は，更新不可能と判断される．

この意図に基づくアプローチの下では，従来のアプローチでは更新不可とされた直積ビューや結合ビューが更新可能になるという意味で，ビューの更新可能性を大いに高めることができる．また，我々は，意図に基づくアプローチの下での直積ビューおよび結合ビューの更新可能性は，ビューが集合意味論の下でのリレーショナル代数で定義されている場合のみならず，バッグ意味論のもとで，バッグ代数で定義されている場合にも成り立つことを報告している [5].

さらに，本アプローチをオープンソースの RDBMS である PostgreSQL 上にプロトタイプを実装し，実際に期待した通りに，ビューの更新可能性を拡張可能であることが確認されている．しかしながら，このプロトタイピングで実装されたアルゴリズムは，想定される変換候補を総当たりに調べるものであり，その全ての変換候補について結果が計算され，その度にビューのマテリアライズが行われるために計算量が膨大となる．そのため，本アプローチを実問題に適用するには計算量の削減の検討が必要となる．

4. 変換候補の生成

上で見たとおり，意図に基づくアプローチでは，ビューに対する更新要求を基に複数の変換候補が生成され，それぞれの計算結果を確認することでビュー更新可能性を判断する．このアプローチを上述の総当たり法で実装した場合には，その変換候補の数だけマテリアライズを伴う結果の計算が必要となる．以下ではまず，実際にどの程度の数の変換候補が生成されるのかを見ていく．

4.1 ビューに対する削除の場合

2 つのバッグ $R(A) = \{a_i(x_i)\}$ $(i = 1...N_a)$ と $S(B) = \{b_j(y_j)\}$ $(j = 1...N_b)$ を考える．ここで，$a_i(x_i)$ はバッグ意味論の表現でバッグにタップル a_i が x_i 個重複して含まれていることを意味している．N_a, N_b はそれぞれ，バッグ R,S に含まれるタップルの種類の数である．これらのバッグ直積ビューを $V = R \times_{bag} S$ と定義すると，その各要素であるタップル (a_i, b_j) の重複度は x_i と y_j の積となる．すなわち，$V = R \times_{bag} S = \{(a_i, b_j)(x_i y_j)\}$ $(i = 1...N_a), (j = 1...N_b)$ と書ける．

このバッグ直積ビュー V に削除要求が発行され，V からタップル集合 $D = \{(a_1^d, b_1^d), (a_2^d, b_2^d), ..., (a_{N_D}^d, b_{N_D}^d)\} = \{(a_i^d, b_i^d)\}$ $(i = 1...N_D)$ が取り除かれようとしているとする（N_D は削除したいタップルの種類の数）．この場合に，考えられうる変換候補がどのようなものかを考える．

なお，バッグ意味論の下では同じ値のタップルが重複して存在することができるが，同じ値のタップル同士は区別することが出来ないものと見なされる．これは，バッグ中の重複した元同士は識別不可能 (indistinguishable) である [12] ためである．これを「バッグの重複元の識別不可能性」と呼ぶことにする．そのため，同じ値のタップルの一部だけが削除されるということは考えず，あるタップルが削除される場合にはそれと同じ

値のタップルは全て削除されるものとする．そのため，上記でバッグ直積ビュー V からタップル (a_i^d, b_i^d) を削除するということは，その重複度に関わらず V から全ての (a_i^d, b_i^d) を削除することを意味する．

まず D の最初のタップル (a_1^d, b_1^d) に注目すると，これを V から削除するには a_1^d を R から削除するか，あるいは，b_1^d を S から削除するか，あるいはその両方を行えば良く，方法は 3 通りあることがわかる．同様に，(a_2^d, b_2^d) を V から削除するには a_2^d を R から削除するか，あるいは，b_2^d を S から削除するか，その両方かの 3 通りである．このことから，V から D に含まれるタップル全てを削除する方法は 3^{N_D} 通りあり，変換候補もこれと同じ数だけ生成されることになる．すなわち，削除するタップルの種類の数の指数オーダとなる．

ビューに対する書換の場合も，ビューに対する削除を行った後に，それに変わる新しいタップルを挿入するの操作と同等であることを考えると，削除の場合と同じだけの変換候補が生成されることとなる．

4.2 ビューに対する挿入の場合

次に，先ほどと同じバッグ $R(A), S(B)$ と，それらからなるバッグ直積ビュー $V = R \times_{bag} S$ に対し，バッグ $I = \{(a_1^I, b_1^I)(k_1), (a_2^I, b_2^I)(k_2), ..., (a_{N_I}^I, b_{N_I}^I)(k_{N_I})\} = \{(a_i^I, b_i^I)(k_i)\}$ $(i = 1...N_I)$ を挿入しようとしているとする．(N_I は挿入したいタップルの種類の数)．この場合に，考えられうる変換候補がどのようなものかを考える．今回は同じ値のタップルの値を重複して挿入できることに注意する．

まず挿入要求のバッグ I の要素のうち，バッグ $R(A)$ に対応する部分だけ見ていく．V に I の最初のタップル $(a_1^I, b_1^I)(k_1)$ を挿入するために a_1^I を R に挿入する必要性を考える．しかし，実際には既に R に a_1^I が含まれている可能性があり，その場合には挿入しなくてもよい．挿入する必要がある場合には，その重複度は高々k_1 までで十分である．すなわち，R に a_1^I を挿入する重複度に関しては 0 から k_1 までの $k_1 + 1$ 通りの可能性がある．

これを I の全てのタップルについて考えると，R への挿入の候補は全部で $\prod_{i=1}^{N_I}(k_i + 1)$ となる．同様に S への挿入の候補は全部も $\prod_{i=1}^{N_I}(k_i + 1)$ であり，全体の候補数はこれらの組み合わせで $\prod_{i=1}^{N_I}(k_i + 1)^2$ となり、そのオーダーはビューに対する削除と同様，挿入するタップルの種類の数の指数オーダとなる．

5. バッグビューの更新可能性判定の連立方程式への形式化

以上のように，考えられる変換候補のオーダの数は指数オーダとなり，これらについて全て検証するという総当たり法を用いる従来のアルゴリズムでは計算量が膨大なものとなる．そこで，この煩雑さを回避し，少ない計算量でビューの更新可能性を判定できる手法を検討した結果，ビューの更新可能性は行の重複度を変数とする非線形連立方程式が解を有するか否かという問題に帰着できることが明らかになった [11]．意図に基づくアプローチでは，ビューのマテリアライズを行うことによりビューへの更新要求の変換可能性を判定するものであるが，そのビューを定義している実バッグの行の重複度を変数とすることで，そのビュー定義のもとでビュー更新要求を満たし得る重複度の組は存在するのか，しないのかをダイレクトに問えばよい，というのが基本的なアイデアである．文献 [11] ではその概要が具体例を用いて示されている．本稿ではこのアイデアの形式化を行う．

5.1 バッグ直積ビューの削除可能性

バッグ直積ビューの削除可能性の判定の形式化を行う．

2 つのバッグ $R(A) = \{a_i(x_i)\}$ $(i = 1...N_a)$ と $S(B) = \{b_j(y_j)\}$ $(j = 1...N_b)$，および，これらのバッグ直積ビュー $V = R \times_{bag} S = \{(a_i, b_j)(x_i y_j)\}$ $(i = 1...N_a), (j = 1...N_b)$ を考える．N_a, N_b はそれぞれ，バッグ R,S に含まれる異なるタップルの種類の数である．

このバッグ直積ビュー V に削除要求が発行され，V からタップル集合 $D = \{(a_1^d, b_1^d), (a_2^d, b_2^d), ..., (a_{N_D}^d, b_{N_D}^d)\} = \{(a_i^d, b_i^d)\}$ $(i = 1...N_D)$ が取り除かれようとしているとする（N_D は削除したいタップルの種類の数）．ユーザの望む結果，すなわち更新意図の外形は V をマテリアライズして，実際に要求された削除を行うことで得られる．これを $W_D = V -_{bag} D = \{(a_i, b_j)(w_{ij})\}$ $(i = 1...N_a), (j = 1...N_b)$ とする．

ここで、何らかの更新の変換 T が存在して，バッグ R, S が更新されたとし、更新後のバッグをそれぞれ $R^d(A) = \{a_i(x_i^d)\}$ $(i = 1...N_a)$ と $S^d(B) = \{b_j(y_j^d)\}$ $(j = 1...N_b)$ と表すと．このとき，更新後のバッグからなるバッグ直積ビューは $V^d = R^d \times_{bag} S^d = \{(a_i, b_j)(x_i^d y_j^d)\}$ $(i = 1...N_a), (j = 1...N_b)$ となる．もしこの変換 T が副作用なく意図した結果を実現するのであれば，$V^d = W_D$，すなわち，以下の方程式がが成立する x_i^d, y_j^d が存在するはずである．

$$x_i^d y_j^d = w_{ij} \ (i = 1...N_a), (j = 1...N_b) \tag{1}$$

w_{ij} の値はマテリアライズにより得た W_D より既知である．さらに，D に含まれるタップルについては削除され W には存在しないはずなので，この式は以下のように書き換えることができる．

$$x_i^d y_j^d = \begin{cases} 0 \ (if (a_i, b_j) \in D) \\ w_{ij} (\geqq 0) \ (otherwise) \end{cases} \tag{2}$$

全ての $1 \leqq i \leqq N_a, 1 \leqq j \leqq N_b$ についてこの式を解いて，整数値 $x_i^d \geqq 0, y_j^d \geqq 0$ の値を全て一意に求めることができれば，このビューは更新可能と判断することができる．

5.2 バッグ直積ビューの挿入可能性

削除可能性の判定と同様に，バッグ直積ビューの挿入可能性の判定の形式化を行う．

次に，先ほどと同じバッグ $R(A), S(B)$ と，それらからなるバッグ直積ビュー $V = R \times_{bag} S$ に挿入要求が発行され，バッグ $I = \{(a_1^I, b_1^I)(k_1), (a_2^I, b_2^I)(k_2), ..., (a_{N_I}^I, b_{N_I}^I)(k_{N_I}\} = \{(a_i^I, b_i^I)(k_i)\}$ $(i = 1...N_I)$ が挿入されようとしているとする．(N_I は挿入したいタップルの種類の数)．この場合も，ユーザの望む結果，すなわち更新意図の外形は V をマテリアライズ

して，実際に要求された挿入を行うことで得られる．これを $W_I = V \cup_{bag} I = \{(a_i, b_j)(w_{ij})\}\ (i = 1...N_a^I), (j = 1...N_b^I)$ とする．N_a^I, N_b^I はそれぞれ，タップル挿入後のバッグ W_I に含まれるドメイン A の値，およびドメイン B の値の種類の数である．

ここで、何らかの更新の変換 T が存在して，バッグ R, S が更新されたとし、更新後のバッグをそれぞれ $R^I(A) = \{a_i(x_i^I)\}\ (i = 1...N_a)$ と $S^I(B) = \{b_j(y_j^I)\}\ (j = 1...N_b)$ と表すと．このとき，更新後のバッグからなるバッグ直積ビューは $V^I = R^I \times_{bag} S^I = \{(a_i, b_j)(x_i^I y_j^I)\}\ (i = 1...N_a^I), (j = 1...N_b^I)$ となる．もしこの変換 T が副作用なく意図した結果を実現するのであれば，$V^I = W_I$，すなわち，以下の方程式が成立する x_i^I, y_j^I が存在するはずである．

$$x_i^I y_j^I = w_{ij} \tag{3}$$

全ての $1 \leqq i \leqq N_a^I, 1 \leqq j \leqq N_b^I$ についてこの式を解いて，整数値 $x_i^I \geqq 0, y_i^I \geqq 0$ の値を全て一意に求めることができれば，このビューは更新可能と判断することができる．

5.3 連立方程式の解法

以上のように，ビューの更新可能性は行の重複度を変数とする非線形連立方程式を解く問題として定式化できることが，形式的に示された．しかしながら，このような非線形連立方程式を解くのは簡単ではない．w_{ij} はマテリアライズにより知ることができるが，$x_i y_j = w_{ij}$ を満たす x_i, y_i の値は網羅的に探索する必要があり，w_{ij} と方程式の数（$N_a \times N_b$）が大きい場合には，その組み合わせは膨大となる．実際に，N_a, N_b はバッグ R,S に含まれるタップルの種類の数であるので大きな値となりうる．また、式 (2) および (3) を生成するためにはバッグに含まれるタップルの種類とその全ての重複度を取得するにはバッグ全体を走査する必要があり，その計算量も無視できない．

そのため，以下では単純に上記の連立方程式を解くことでビューの更新可能性を判断するのではなく，別の方法を検討する．具体的には，ここでの形式化の結果を利用して，探索範囲の「枝刈り」を行うことで，従来の総当たり法よりも計算量でビュー更新可能性判断を行う方法を検討する．

6. 計算量削減方法の検討

4. 章で述べたとおり，現行のアルゴリズムではビュー更新可能性を判断するには，更新要求のタップルの種類の数 N に対して K^N のオーダーの数の変換候補について，マテリアライズを伴う検査を行う必要がある．以下では，前章の形式化を元に，検査の対象となる変換候補の数を削減する方法を検討する．

6.1 ビューの削除可能性判断の効率化

バッグ直積ビューの削除可能性の判定の効率化のため，制約として以下の仮定をおくことにする．

- ビューに対する削除は基底バッグの削除に変換される．

すなわち，5. に定義したバック直積ビュー V への削除要求は，基底バッグ R, S への削除に変換されるものとする．この制約は理論的に導かれた要件ではないが直観的に道理にかなっているものとして文献 [2] で採用されているものであり，本論文でもこれを採用する．この制約の下では，更新後のバッグ R^d, S^d のタップルの重複度は，そのタップルが削除されたのならば 0，削除されなかったのならば元のバッグ R, S と同じ値をとることになる．これにより，式 (2) は以下のように書き換えられる．

$$\begin{cases} x_i^d = 0\ or\ y_j^d = 0\ (if (a_i, b_j) \in D) \\ x_i^d = x_i\ and\ y_j^d = y_j\ (if (a_i, b_j) \in W_D) \end{cases} \tag{4}$$

式 (4) の上段は，5. 章の「(a_i^d, b_i^d) を V から削除するには a_i^d を R から削除するか，あるいは，b_i^d を S から削除するか，その両方かの 3 通り」という記述に対応する．では，どのタップルを削除するべきか，その手がかりは下段の式を使って得ることができる．もし，$(a_i^d, *)$（*は任意のドメイン B のタップル）のようなタップルが 1 つでも W_D に存在するならば，x_i^d は 0 ではなく，すなわち a_i^d は削除されず，b_j^d が削除されなければならない．逆に，もし，$(*, b_j^d)$（*は任意のドメイン A のタップル）のようなタップルが 1 つでも W_D に存在するならば，y_j^d は 0 ではなく，すなわち b_j^d は削除されず，a_i^d が削除されなければならない．また，あるタップルが削除されない（または、される）ことが分かれば，この情報はその後の変換候補の更新可能性判断でも使用可能であり，その時には W_D に対する検索は必要ない．

以上のように，ビューの削除可能性の判断を全ての変換候補についてマテリアライズを実行せずに行うことができることが示された．マテリアライズの代わりに W_D の検索が伴うが，これは常に行われるわけではなく，またマテリアライズによる検査より計算量の少ない処理であるため，ここで述べたアプローチによりアルゴリズムが効率化することが期待される．

6.2 ビューの挿入可能性判断の効率化

バッグ直積ビューの挿入可能性の判定の効率化のため，制約として以下の仮定をおくことにする．

- ビューに対する挿入は基底バッグの挿入に変換される．

すなわち，5. に定義したバック直積ビュー V への挿入要求は，基底バッグ R, S への挿入に変換されるものとする．ここで，R, S に挿入されるタップルをそれぞれ $dR = \{a_i(dx_i)\}, dS = \{b_j(dy_j)\}$ とする $(dx_i \geqq 0, dy_i \geqq 0)$．このとき，更新後のバッグは $R^I = R \cup_{bag} dR = \{a_i(x_i + dx_i)\}, S^I = S \cup_{bag} dS = \{b_j(y_j + dy_j)\}$，更新後のバッグからなるバッグ直積ビューは $V^I = R^I \times_{bag} S^I = (R \times_{bag} S) \cup_{bag} (R \times_{bag} dS) \cup_{bag} (dR \times_{bag} S) \cup_{bag} (dR \times_{bag} dS) = \{(a_i, b_j)(x_i y_j + x_i dy_j + dx_i y_j + dx_i dy_j)\} = V \cup_{bag} \{(a_i, b_j)(x_i dy_j + dx_i y_j + dx_i dy_j)\}$ となる．$V^I = V \cup_{bag} I$ であるから $I = \{(a_i, b_j)(k_{ij})\} = \{(a_i, b_j)(x_i dy_j + dx_i y_j + dx_i dy_j)\}$ と書くことができ，以下の式が成立する．

$$k_{ij} = x_i dy_j + dx_i y_j + dx_i dy_j \tag{5}$$

k_{ij} は挿入するタップル (a_i, b_j) の重複度であり，更新要求から取得可能な既知な値である．等式が成り立つためには，dx_i, dy_j は k_{ij} の値を超えてはならない．これは 5. 章の「R に a_1^I を挿入する重複度に関しては 0 から k_1 までの $k_1 + 1$ 通り

の可能性がある」という記述に対応する．また，x_i, y_j は a_j, b_j の重複度であり，R, S を検索することで取得可能な既知な値である．この値を利用すると dx_i, dy_j が取りうる値をさらに制限することができる．

まず，タップル (a_i^I, b_j^I) が V に挿入される前提から $k_{ij} > 0$ である．そのため，少なくとも $x_i dy_j, dx_i y_j, dx_i dy_j$ のいずれかが正の数でなければならない．もし，a_i^I が R に存在しない場合には $x_i = 0$ となるが，このとき $k_{ij} > 0$ を満たすためには，$dx_i > 0$ でなければならない．すなわち，R に 1 つ以上の a_i^I を挿入する必要がある．同様に，b_j^I が S に存在しない場合には，S に 1 つ以上の b_j^I を挿入する必要がある．これは文献 [11] でも示されているヒューリスティックである．一方で，$x_i dy_j$ が k_{ij} を超えてはならないことから，dy_j の値は高々k_{ij}/x_i であることがわかる．同様に dx_i の値は高々k_{ij}/y_j である．

以上のように，R, S に対して検索を行うことで，dx_i, dy_j の値の範囲を限定し，ビューの挿入可能性の判断を行う際の変換候補を削減できることが示された．

7. おわりに

本稿では，ビュー更新問題を解決する意図に基づくアプローチと，具体的なアルゴリズムである「更新意図の外形的推測」を概説し，現行のアルゴリズムでは多数生成される変換候補についてマテリアライズを実施するために，計算量が問題となることを示した．また，この問題を解決に向けて，バッグ意味論における直積ビューと結合ビューの更新問題を非線形連立方程式の形に形式化できることを示し，この知見を元に計算量を削減する方法を検討した．今後は，本稿で検討した内容に基づき，より効率の良いビュー更新アルゴリズムを開発し，それを PostgreSQL 上に実装し，性能検証を行っていく．

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

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，長田 悠吾，石井 達夫，“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－,” DEIM Forum 2017 会議録, H3–5, 2017 年 3 月.

[6] Y. Nagata and Y. Masunaga, “Extending View Updatability by a Novel Theory –Prototype Implementation on PostgreSQL–,” PGCon 2017-The PostgreSQL Conference, Ottawa, Canada, May 25–26, 2017.

[7] 長田悠吾，石井達夫，増永良文，“意図に基づくアプローチによる更新可能ビューの拡張― PostgreSQL におけるプロトタイピング ―,” WebDB Forum 2017 会議録，DBS–165, No.14, 2017 年 9 月.

[8] Y. Masunaga, Y. Nagata and T. Ishii, “Extending the View Updatability of Relational Databases from Set Semantics to Bag Semantics and Its Implementation on PostgreSQL,” Proceedings of the 12th International Conference on Ubiquitous Information Management and Communication (ACM IMCOM 2018), 8–3, Langkawi, Malaysia, 2018.

[9] 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.

[10] 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.

[11] 増永 良文, 長田 悠吾, 石井 達夫，“意図に基づくアプローチのもとでの SQL ビューの更新可能性―非線形連立方程式問題への帰着―,” DEIM Forum 2018 会議録, G1–3, 2018 年 3 月.

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