マテリアライズドビューの更新問題：
ローカルで更新可能なマテリアライズドビュー

増永 良文† 長田 悠吾‡ 星合 拓馬‡ 石井 達夫‡
†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
‡SRA OSS, Inc. 日本支社 〒171-0022 東京都豊島区南池袋 2-32-8
E-mail: †masunaga.yoshifumi@ocha.ac.jp ‡{nagata, hoshiai, ishii}@sraoss.co.jp

あらまし マテリアライズドビューは，(仮想的)ビューと違って，更新の対象とはされずに専ら読み取り専用でのサポートがなされている現状がある．この理由としては，マテリアライズドビューはデータベースに格納されているビューなのでインデックスを張れるなどビューに比べて高速な問合せ処理が可能な点を挙げられる．一方で，マテリアライズドビューの更新可能性はビューの更新可能性と同様に形式化できるものの，マテリアライズドビューが更新可能な場合には，更新されたマテリアライズドビューとデータベースが整合するようにデータベースの実テーブルを適当なタイミングで更新する必要があり，そのためのコストが嵩むことになる点を挙げられよう．しかしながら，何らかの形でマテリアライズドビューの更新が許されるのであれば，ユースケースの拡大に繋がることは言をまたない．本稿では，マテリアライズドビューの更新に対する上記のジレンマを解決する一つの方策として「ローカルで更新可能なマテリアライズドビュー」という新規概念を導入し，その形式化を行った結果を報告する．

キーワード ビュー，マテリアライズドビュー，マテリアライズドビューの更新問題，ローカルで更新可能なマテリアライズドビュー，意図に基づくアプローチ

1 はじめに

マテリアライズドビューは体現化されている，つまりその定義だけではなくビューを定義している問合せ(query)が評価され，その結果がデータベースに格納されているビューである．従って，インデックスを張ることも可能なので，マテリアライズドビューに対して発行された問合せは(仮想的)ビュー，つまりその定義だけが格納されているビュー，に対して発行された場合と比較して高速に処理可能である．一方で，マテリアライズドビューを更新しようとした場合，その更新可能性はビューの更新可能性と同様に形式化できるものの,マテリアライズドビューが更新可能な場合には，更新されたマテリアライズドビューとデータベースが整合するようにデータベースの実テーブルを適当なタイミングで更新する必要があり，そのためのコストが嵩むことになる点を挙げられよう．

現在，マテリアライズドビューは集約処理を施したデータを格納したデータウェアハウスなど，問合せが主体となるデータベースの利用環境で用いられることが多い[4]．また，商用あるいはオープンソースソフトウェア(OSS)のリレーショナル DBMS の多くがマテリアライズドビューをサポートしているものの，想定されているユースケースは上記のような問合せ業務が主体であるようで，マテリアライズドビューの更新機能をサポートしているリレーショナル DBMS は見当たらない．かつて，Oracle Database(以下，Oracle)は，Oracle 12c リリース 1(12.1)までは 3 種のマテリアライズドビュー，即ち，読取り専用，更新可能，書込み可能なマテリアライズドビューをサポートしていたことが報告されているが，12.2 以降は読取り専用のみのサポートとなっている[8, 9, 10]．当時，読取り専用，つまり問合せ専用に加えて，更新可能と書込み可能をサポートした狙いは，マテリアライズドビューを使用したデータのレプリケーション機能のサポートにあったようで，更新可能と書込み可能なマテリアライズドビューに対して次のような説明がなされている (引用は原文ママ．以下，一部は中略，後略あり)．

『更新可能マテリアライズド・ビューは、表のコピーへの読取り/書込みアクセスを提供します。更新可能マテリアライズド・ビューを使用して、アプリケーションおよびユーザーは、表と表のコピーの両方を変更でき、これらの変更をある時点で同期できます。たとえば、更新可能マテリアライズド・ビューは、通常は地域のオフィスに製品カタログを定期的に配布するため、また営業員が顧客サイトから注文を行えるようにするために使用されます。』[8]

『(前略)ユーザーは書込み可能マテリアライズド・ビューに対して DML 操作を実行できますが、マテリ

アライズド・ビューをリフレッシュしたときに、この変更はマスターにプッシュされず、マテリアライズド・ビュー自体からも変更が失われます。(後略) (注意: 書込み可能マテリアライズド・ビューはほとんど使用されないため、マテリアライズド・ビューに関するドキュメントのほとんどは、読取り専用および更新可能マテリアライズド・ビューのみを説明しています。)』[9]

補足すると，Oracle 12.2 以降は，12.1 まであった advanced replication という機能のサポートの全面的終了により，更新可能マテリアライズドビューや書込み可能マテリアライズドビューのサポートは終了し，読取り専用マテリアライズドビューだけがサポートとなったようである[10].

さて，マテリアライズドビューは国際標準リレーショナルデータベース言語(以下，SQL 標準）の仕様とはなっていない．ビューが SQL 標準の仕様となっているのに，マテリアライズドビューがそうでない理由は定かではないが，そのような事情からマテリアライズドビューの定義法や利用法についてはリレーショナル DBMS のベンダ毎の取組に任されているのが現状である．先述の通り，更新可能なマテリアライズドビューをサポートしているリレーショナル DBMS は現在のところ見当たらないが，問合せ処理が高速であるマテリアライズドビューに更新が許されれば，ユースケースの拡大が期待できることは大いに考えられる．本稿では，更新可能なマテリアライズドビューという概念に加えて，体現化されているというマテリアライズドビューの特徴に着目することにより,「ローカルで更新可能なマテリアライズドビュー」という新しい概念を導入し，それを形式化した結果を報告する．

以下,本稿は,マテリアライズドビューとは(第 2 章), 問題提起(第 3 章)，更新可能なマテリアライズドビュー(第 4 章)，ローカルで更新可能なマテリアライズドビュー(第 5 章)，おわりに(第 6 章)と続く．

2 マテリアライズドビューとは

SQL 標準にマテリアライズドビューの仕様はなく，従って，マテリアライズドビューの定義は商用あるいは OSS のリレーショナル DBMS のベンダ任せとなっていることは先述した．例えば，PostgreSQL の定義構文は次の通りである(パラメタの説明は省略する).

CREATE MATERIALIZED VIEW t*able_name*
[(*column_name* [, ...])]
[WITH (*storage_parameter* [= *value*] [, ...])]
[TABLESPACE *tablespace_name*]
AS *query*
[WITH [NO] DATA]

ちなみに，Oracle では次のようである．

CREATE MATERIALIZED VIEW *view-name*
BUILD [IMMEDIATE | DEFERRED]
REFRESH [FAST | COMPLETE | FORCE]
ON [COMMIT | DEMAND]
[[ENABLE | DISABLE] QUERY REWRITE]
AS
SELECT ...;

定義構文の詳細はこのようにベンダにより異なるが，その骨子は次の通りである．

CREATE MATERIALIZED VIEW *table_name*
AS *query*;

ビューとマテリアライズドビューの違いは，構文的には MATERIALIZED が入っているか否かである．生成されたマテリアライズドビューはその定義を変更したり,あるいはそれを削除したりすることが出来るが，その構文の概略は次の通りである．

ALTER MATERIALIZED VIEW *table_name*;
DROP MATERIALIZED VIEW *table_nam*e;

ここで，単純なマテリアライズドビューの例を 1 つ示しておく．

【例題 1】(マテリアライズドビューの定義例)
実テーブル *supply* と *demand*，それらの自然結合で定義されるマテリアライズドビューを *sd* とする．

CREATE MATERIALIZED VIEW *sd* AS
SELECT*FROM *supplier* NATURAL JOIN *demand*;

実テーブル *supply* と *demand* の実体を次の通りとする(キーをアンダーラインで表示).

supply

<u>*supplier*</u>	<u>*part*</u>
s1	p1
s1	p2
s2	p1

demand

<u>*part*</u>	<u>*demander*</u>
p1	d1
p2	d2

このとき，具現化された *sd* は次の通りである．

sd

<u>*supplier*</u>	<u>*part*</u>	<u>*demander*</u>
s1	p1	d1
s1	p2	d2
s2	p1	d1

3 問題提起

3.1 マテリアライズドビューの更新とは

例題 1 に示したマテリアライズドビュー*sd* を例にとり，マテリアライズドビューの更新とは何かを考察することから始める．更新には，削除(delete)，挿入(insert)，書換(update or rewrite)があるが，例えば *sd* に対して次の削除操作 ***d*** が発行されたとしよう．

d : DELETE FROM *sd* WHERE
supplier=s1 AND *part* = p2 AND *demander*= d2;

さて，ここで提起したい問題は次の通りである．

“マテリアライズドビュー*sd* に対して発行された削除操作 ***d*** を受理するべきか否か”

解答するにあたって，次の 2 つの立場が考えられる．

(a) 従来のビュー更新問題の結果に従い受理するか否かを決めればよい

(b) そうではなく，マテリアライズドビューの更新問題を改めて考察し，その結果に従って決めるべき．

(a)はマテリアライズドビューも仮想的ビューも共にビューであることに間違いはないので，これまで仮想的ビューの更新問題を論じて得られた結果を素直にマテリアライズドビューの更新問題に適用し，その結果で判断してよいのではないか，つまり *sd* を仮想的ビューと捉えた時，それが ***d*** を受理できるのであれば受理するべきであるしそうでなければ拒否する，という考え方である．

一方，(b)は，確かに両者はビューであることに違いはないが，マテリアライズドビューは具現化されているが故に，そこにマテリアライズドビューならではの特殊性が潜んでいるのではないか．従って，それをきちんと考慮し，そのうえで結論を導くべきではないか，という立場である．

では，マテリアライズドビューを更新しようとすると，どのような問題が生じるのであろうか?それを例題 1 で示した例を再考することで示す．

【例題 2】実テーブル *supply* と *demand*，マテリアライズドビュー*sd*，それに対して発行された削除操作 ***d*** は上述の通りとする．

まず，***d*** を実行して得られた結果，これを sd_{new} と書く，は次の通りである．

sd_{new}

supplier	*part*	*demander*
s1	p1	d1
s2	p1	d1

さて，sd_{new} は *sd* が ***d*** により更新された結果であるが，問題は，この削除操作の結果を受け入れてよいか，否かである．確かに，マテリアライズドビューは具現化されているので，どのような DML 操作も適用可能である．従って，*sd* に ***d*** を適用すれば sd_{new} が得られるのは当然である．しかしながら，マテリアライズドビュー*sd* は実テーブル *supply* と *demand* の自然結合ビューとして定義され具現化された結果であるという事実は厳然とある．これは *sd* に課せられた“制約”と考えてよい．従って，*sd* に対する ***d*** が受理されるためには，$sd_{new}=supply_{new}*demand_{new}$ となる $supply_{new}$ や $demand_{new}$ が存在するべきなのではないかと考えられる．実際 $supply_{new}$ と $demand_{\sout{new}}$ を次のようにすれば，その条件を満たせうる．この結果，マテリアライズドビューとデータベース間のデータの整合性が保たれる．そして，もしそうならば，***d*** を受理してよいのではないかと考えられる．

$supply_{new}$

supplier	*part*
s1	p1
s2	p1

$demand_{\sout{new}}$

part	*demander*
p1	d1

では，本当にこれで *sd* に対する ***d*** を受理してよいと結論してよいであろうか?

この結論に対する反論は容易に示すことができる．つまり，容易に検証できるように，次が成立することが分かる．

$$sd_{new}=supply_{new}*demand$$
$$sd_{new}=supply*demand_{new}$$

つまり，マテリアライズドビュー*sd* に対して発行された削除操作 ***d*** を実現できる実テーブル *supply* と *demand* に対する更新操作は計 3 つあるということである．(上記 $sd_{new}=supply_{new}*demand_{new}$ の場合は $T_3(\boldsymbol{d})$に該当する)

$T_1(\boldsymbol{d})$: DELETE FROM *supply* WHERE *supplier* = s1 AND *part*= p2;

$T_2(\boldsymbol{d})$: DELETE FROM *demand* WHERE *demander* = d2 AND *part* = p2;

$T_3(\boldsymbol{d})$: BEGIN; DELETE FROM *supply* WHERE *supplier*=s1 AND *part* = p2; DELETE FROM *demand* WHERE *demander* = d2 AND *part* = p2; END;

従って，問題はこの曖昧性をどう解釈するかである．言うまでもなく，従来のビューの更新可能性の観点からは，この曖昧性に依り，*sd* へ発行された削除操作 ***d*** は受理できない．何故ならば，確かに $T_1(\boldsymbol{d})$～$T_3(\boldsymbol{d})$のいずれを採用しても ***d*** を実現できるが，$T_1(\boldsymbol{d})$～$T_3(\boldsymbol{d})$が現実世界で担っている意味が全く異なっており，その曖昧性を解消できないからである[5].

つまり，この例題 2 で行った議論をまとめてみると，次の 2 つの立場が鮮明になる．

“*sd* に対して発行された削除操作 ***d*** を実行した結果である sd_{new} を実現できる実テーブルの更新操作は存在するものの複数あり，そのうちのどれを採ればよいか，その曖昧性を解消できない以上，***d*** を受理するべきではない．”(前記，(a)に対応)

“いや，確かに曖昧性はあるものの，いずれを採っても sd_{new} が supply と demand の自然結合ビューであるという制約を満たしているので，***d*** を受理してよいのではないか．” (前記，(b)に対応)

そこで，我々は，この 2 つの立場を十分に認識した

うえで，マテリアライズドビューの更新問題を次のように形式化した．

3.2 マテリアライズドビューの更新問題の形式化

上述の 2 つの立場は，それぞれに意味があると考えられる．そこで，マテリアライズドビューの更新について 2 つの概念を導入する．

(1) 更新可能なマテリアライズドビュー

(2) ローカルで更新可能な(locally updatable)マテリアライズドビュー

(1)は，前節(a)の立場に立った定義である，一方，(2)は前節(b)の立場に立った定義である．つまり，前者は，マテリアライズドビューといえども，その更新は従来のビューの更新と同じ考え方でアプローチするべきであるという主張である．一方，後者はビューが具現化されているがゆえに意味があるとする立場に立った概念規定である．ここで，ローカルの意味を再確認しておくと，マテリアライズドビューに対して発行された更新はデータベースには反映しない，あるいはできないが，このマテリアライズドビューだけに限定すれば許される，という意味である．「ローカルな更新可能性」を許すことにより，実際のデータベース更新は実行されないが，意味的には問題を起こさないマテリアライズドビューへの更新は許されるので，それに基づいたアプリケーションの検証などが可能となるであろう．

以下，これら 2 つの概念を形式化する．まず，更新可能なマテリアライズドビューの定義を形式的に与えると次の通りである．

【定義 1】(更新可能なマテリアライズドビュー)
マテリアライズドビュー m_v が更新操作 $\boldsymbol{u}$ のもとで更新可能であるとは，図 1 の可換図式が成立するとき及びそのときのみをいう．ここで，$\exists!\boldsymbol{v}$ は DB に対する更新 $\boldsymbol{v}$ がただ 1 つ存在することを表す(曖昧性の排除)．

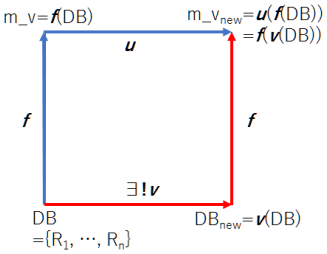

図 1. 更新可能なマテリアライズドビュー

次に，ローカルな更新可能性の定義を示す．

【定義 2】(ローカルで更新可能なマテリアライズドビュー)
マテリアライズドビュー m_v が更新操作 $\boldsymbol{u}$ のもとでローカルで更新可能であるとは，図 2 の可換図式が成立するとき及びそのときのみをいう．ここで，$\exists\boldsymbol{v}$ は DB に対する更新 $\boldsymbol{v}$ が存在することを表す．

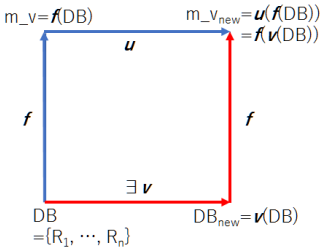

図 2. ローカルで更新可能なマテリアライズドビュー

改めて記すまでもないが，2 つの更新可能性の違いは，唯一，$\boldsymbol{v}$ の存在が一意であるか，そうでないかにある．

先述したが，現在，多くの商用あるいは OSS のリレーショナル DBSM でマテリアライズドビューを定義できるが，それらがサポートするマテリアライズドビューは問合せ専用である．

以下，これら 2 つのマテリアライズドビューの更新可能性をさらに詳しく論じる．

4 更新可能なマテリアライズドビュー

4.1 バッグとバッグ代数とビュー

幾つか基礎的事項を与える．本稿では，データベースは一般に SQL が準拠する“バッグ意味論”(bag semantics) に従っているとする．バッグ意味論はリレーショナルデータモデルが準拠している集合意味論 (set semantics) の拡張となっている．従って，リレーショナル代数に代わってバッグ代数がビューを形式的に定義するときに使われる．

まず，バッグ(マルチ集合(multi-set)ともいう)の定義を与えておく(以下，$\vee$，$\wedge$，$\neg$，$\Rightarrow$，iff はそれぞれ論理和，論理積，論理否定，含意 (logical implication)，if and only if を表す)．

【定義 3】(バッグ) $\boldsymbol{R}(A_1, A_2, \ldots, A_n)$をリレーションスキーマとする．$\Omega_{\boldsymbol{R}}=\{A_1, A_2, \ldots, A_n\}$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 は次のように定義される．

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

ここに，k_i は R 中に生起する行(row)t_i の重複度 (multiplicity) を表す．このとき，重複度関数 m を $m(R, t_i)=k_i$ と定義する．バッグ $R=\{t_1(k_1), t_2(k_2), \ldots, t_p(k_p)\}$の濃度 (cardinality) を$|R|$と表し，$|R|=\sum_{i=1,..,p}k_i$ と定義する．また，$(\forall t_i(k_i)\in R)(\forall t_j(k_j)\in R)(t_i=t_j \Rightarrow k_i=k_j)$とする．これは，マルチ集合論の前提となっている「重複元の識別不可能性原理」[1]によるものである．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 と書くことも多い．また $t_i(0)$は t_i が存在していないことを表す．次にバッグ代数を定義する[2, 3]．

【定義 4】(バッグ代数)バッグ代数は重複行除去, 和, 差, 共通, クロス結合 (＝直積), 射影, θ-選択, θ-結合, 集約演算子からなる. その定義を表 1 に与える. ここに, 一般にバッグ R や S を R={$t_1(k_1)$, $t_2(k_2)$, …, $t_p(k_p)$}, S={$u_1(\ell_1), u_2(\ell_2), \ldots, u_q(\ell_q)$}とし, 和, 差, 共通演算では p=q でかつ R と S は和両立と仮定する.

表 1 バッグ代数の演算子とその定義

演算子	定義
重複行除去 δ	$\delta(R)=\{t_1, t_2, \ldots, t_p\}$
和 $R\cup_{bag}S$	$R \cup_{bag} S = \{t_i(k_i)\|t_i \in \delta(R) - \delta(S)\} \cup \{t_i(k_i+\ell_j)\|t_i=u_j \in \delta(R)\cap\delta(S)\} \cup \{u_j(\ell_j)\|u_j \in \delta(S)-\delta(R)\}$. 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.
差 $R-_{bag}S$	$R -_{bag} S = \{t_i(k_i)\|t_i \in \delta(R) - \delta(S)\} \cup \{t_i(\max(k_i-\ell_j, 0))\|t_i=u_j\in\delta(R)\cap\delta(S)\}$. ここに, $\max(k_i-\ell_j, 0)$は $k_i-\ell_j$ と 0 の大きな値の方をとる演算.
共通 $R\cap_{bag}S$	$R\cap_{bag}S=t_i(\min(k_i, \ell_j))\|t_i=u_j\in\delta(R)\cap\delta(S)\}$. 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.
クロス結合 $R\times_{bag}S$	$R\times_{bag}S = \{(t_i, u_j)(k_i \times\ell_j)\|t_i \in \delta(R) \wedge u_j \in \delta(S)\}$.
射影 $R[X]_{bag}$	$R[X]_{bag}=\{x(k)\|(\exists\ t_i \in \delta(R))(x=t_i[X]) \wedge k=\sum_j k_j$ such that $t_j[X]=x\}$. ここに, $X\subseteq\Omega_R$.
θ-選択 $R[Ai\,\theta\,Bj]_{bag}$	$R[Ai\theta Aj]_{bag}=\{t_i(k_i)\|t_i\in\delta(R)\wedge t_i[A_i]\theta t_i[A_j]\}$. ここに, R の属性 A_i と A_j は θ-比較可能.
θ-結合 $R[Ai\,\theta\,Bj]_{bag}S$	$R[A_i\theta B_j]_{bag}S=\{(t_i, u_j)(k_i\times\ell_j)\|t_i\in\delta(R)\wedge u_j\in\delta(S)\wedge t_i[A_i]\theta u_j[B_j]\}$. ここに, $R.A_i$ と $S.B_j$ は θ-比較可能.
集約 $\gamma_{A,\ f(B)}(R)$	$\gamma_{A,\ f(B)}(R)=\{(t_i[A],\ f(B))\|t_i \in \delta(R) \wedge f(B)$=aggregation f of all $t_i(k_i)$ such that $t_i[A]=t_i[A]\}$. ここに, A と B は R の属性.

ここでバッグ代数に関して幾つか注意点を記す. まず, $t(k)\in R$ iff $t\in\delta(R)\wedge m(R, t)=k$ である. なお, 上記 9 個の演算はお互いに独立ではなく, それらの定義から導かれるように, 例えば, バッグ R と S の θ-結合 $R[A_i\ \theta\ B_j]_{bag}S$, ここに属性 $R.A_i$ と $S.B_j$ は θ-比較可能とする, はクロス結合演算と θ-選択演算を使って, $R[A_i\ \theta\ B_j]_{bag}S=(R\times_{bag}S)[R.A_i\ \theta\ S.B_j]_{bag}$ と書ける. また, バッグ $R(A, B)$ と $S(B, C)$ の自然結合は, $R*_{bag}S=\{(t_i, u_j[C])(k_i\times\ell_j)\|t_i\in\delta(R)\wedge u_j\in\delta(S)\wedge t_i[B]=u_j[B]\}$であるが, $R*_{bag}S=(R[R.B=S.B]_{bag}S)[\{R.A, R.B, S.C\}]_{bag}$ と等結合と等選択演算を使って書ける. 独立でない演算子を許容しているのは, それらの有用性による.

従って, バッグ意味論の下でのビューをバッグ代数を使って次のように定義することが出来る. ここに, 実バッグ(base bag)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.

【定義 5】(バッグ代数を用いたビュー)

1. 実バッグ R はビューである.
2. V をビューとするとき, $\delta(V)$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, $V_1\cup_{bag}V_2$, $V_1-_{bag}V_2$, $V_1\cap_{bag}V_2$ 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, $V_1\times_{bag}V_2$ はビューである. これをクロス結合(＝直積)ビューという.
5. V をビューとするとき, $V[X]_{bag}$ 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, $V[A_i\,\theta\,A_j]_{bag}$ 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, $V_1[A_i\,\theta\,B_j]_{bag}V_2$ 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. V をビューとするとき, $\gamma_{A,\ f(B)}(V)$はビューである. これを集約関数 f の下での集約ビューという.
9. 1. から8. で定義されるもののみが, ビューである.

4.2 更新可能なマテリアライズドビュー

更新可能なマテリアライズドビューは図 1 の可換図式を満たさないといけない. これは(仮想的)ビューの更新可能性と概念的に同一であり, 集合意味論の下での結果が筆者らの「意図に基づくアプローチ」で示されている[5]. それを基に, バッグ意味論における 8 つの基本ビューへの更新の変換可能性を検証した結果は表 2 に示す通りとなる. 従って, それが更新可能なマテリアライズドビューを規定していることになる.

表 2 バッグ意味論の下での 8 つの基本ビューへの更新の変換可能性

	重複行除去	和	差	共通	クロス結合	射影	θ-選択	θ-結合
D	○	○†1	×	×	○†1	○	○†3	○†1,3
I	×	×	△	△	○	○†2	○	○
R	×	×	×	×	○†1,4	○†2	○†3	○†1,3,4

D：削除, I：挿入, R：書換. ○：変換可能, □：場合により変換可能, ×：変換不可, △：バッグ意味論の下で変換不可, ただし集合意味論の下では変換可能, †1：重複元の識別性不可能原理[1], †2：キー保存（集合意味論の下での条件), †3：同種変換の原理, †4：書換操作の両立.
(†2～†4 は[5]で与えられている通り)

なお, 更新可能なマテリアライズドビューへの更新とビューに対する更新の違いは, その可能性にあるのではなく, マテリアライズドビューが更新可能な場合には, 更新されたマテリアライズドビューとデータベースが整合するようにデータベースの実テーブルを適当なタイミングで更新する必要があり, そのためのコストが嵩むことになる点を挙げられよう.

5 ローカルで更新可能なマテリアライズドビュー

5.1 マテリアライズドビューであることの制約

マテリアライズドビューは自由に生成することが出来るが，生成されたマテリアライズドビューにはその定義に即した制約が伴うことになる．典型例として，例題 1 で示した自然結合ビュー*sd* に課せられた制約とは何かについて考察する．

【例題 3】(体現化された自然結合ビューの制約)

まず，$R(A, B)$, $S(B, C)$の自然結合ビュー$R*_{bag}S$ の定義は次の通りであった．

$$R*_{bag}S= (R[R.B=S.B]_{bag}S)[\{R.A, R.B, S.C\}]_{bag}$$

従って，$(a, b, c)(k) \in R*_{bag}S$ iff $(a, b, b, c)(k) \in R[R.B=S.B]_{bag}S$ である．一方，等結合の定義は一般に $R[A_i=B_j]_{bag}S = \{(t_i, u_j)(k_i \times \ell_j) | t_i \in \delta(R) \wedge u_j \in \delta(S) \wedge t_i[A_i]=u_j[B_j]\}$．ここに，$R.A_i$ と $S.B_j$ は θ-比較可能であったから，この例題では$(a, b, b, c)(k_i \times \ell_j) \in R[R.B=S.B]_{bag}S$ iff $(a, b)(k_i) \in R \wedge (b, c)(\ell_j) \in S$ となる．従って，自然結合 $R(A, B)*_{bag}S(B, C)$については，$(a, b, c)(k_i \times \ell_j) \in R(A, B)*_{bag}S(B, C)$ iff $(a, b)(k_i) \in R \wedge (b, c)(\ell_j) \in S$ となる．換言すると，これが体現化された自然結合ビュー$R(A, B)*_{bag}S(B, C)$が満たすべき制約である．

従って，例題 1 で示した自然結合ビュー*sd* に課せられている制約とは，$R(A, B)$=*supply*(*supplier, part*)，$S(B, C)$=*demand*(*part, demander*)なので，次のようになる．

$(s, p, d)(k_i \times \ell_j) \in sd$ iff

$(s, p)(k_i) \in supply \wedge (p, d)(\ell_j) \in demand$

5.2 バッグ MVD の導入

ここで．バッグ意味論の下での自然結合ビューの制約についての議論を形式化する．

従来，集合意味論の下で，リレーション $R(A, B, C)$ がその 2 つの射影，$R[A, B]$と $R[B, C]$に情報無損失分解される，つまり $R(A, B, C)=R[A, B]*R[B, C]$が成立するための必要かつ十分条件は，$R$ に多値従属性(multiple dependency, MVD) $B \rightarrow\rightarrow A|C$ が存在することであった．この議論をバッグ意味論に拡張してみる：

Tを一般にバッグとする．このとき，$T(A, B, C)=R(A, B)*_{bag}S(B, C)$が成立するための必要かつ十分かつ十分条件は $T(A, B, C)$に“バッグ多値従属性”(bag MVD, BMVD) $B \Rightarrow\Rightarrow A|C$ が存在すること．

実際，簡単な例を示すと次のようである．

【例題 4】(BMVD)　$T(A, B, C)$を次に示す通りとする．

T

A	B	C
1	2	3
1	2	3
1	2	3
1	2	3
2	3	4

つまり，T={(1, 2, 3)(4), (2, 3, 4)}は，4=1×4, 4=2×2, 4=4×1 の 3 通りの組合せがあるので，バッグ意味論の下での T の情報無損失分解は次の 3 通りあることになる．

R	
A	B
1	2
1	2
1	2
1	2
2	3

S	
B	C
2	3
3	4

R	
A	B
1	*2*
1	2
2	3

S	
B	C
2	*3*
2	3
3	4

R	
A	B
1	2
2	3

S	
B	C
2	3
2	3
2	3
2	3
3	4

従って，体現化された自然結合ビュー$T(A, B, C)=R(A, B)*_{bag}S(B, C)$に対する更新操作 ***u*** が発行されたとき，***u*** がローカルで受理可能か否かは，T に BMVD $B \Rightarrow\Rightarrow A|C$ が存在しているか否かで決定できる．

5.3 バッグ MVD が存在するか否かの検証法

バッグ意味論の下で，体現化された自然結合ビュー$T(A, B, C)=R(A, B)*_{bag}S(B, C)$が与えられたとき，$T$ で BMVD $B \Rightarrow\Rightarrow A|C$ が成立しているか否かを検証する方法を制約充足問題(constraint satisfaction problem, CSP)[11]として捉え，解決する．これはビューが体現化されているが故に有効な手法である．

さて，$T(A, B, C)=R(A, B)*_{bag}S(B, C)$を体現化された自然結合ビューとする．$T$ に更新操作 ***u*** が発行されて，その結果 T が T_{new} になったとする．T_{new} に BMVD $B \Rightarrow\Rightarrow A|C$ が存在するかどうかを CSP の一種として知られている整合ラベリング問題(consistent labeling problem, CLP)で形式化し，それを解くための手法として知られる併合法(merging)[7]を使って解いてみる．それを例題で示す．

【例題 5】$R(A, B)$={(1, 2)(2), (2, 2)(1)}と $S(B, C)$={(2, 3)(1), (2, 4)(1)}を実バッグ，$T(A, B, C)$をそれらより自然結合演算を用いて生成されたマテリアライズドビューとする．体現化された Tは次に示す通りである．

T

A	*B*	*C*
1	2	3
1	2	3
1	2	4
1	2	4
2	2	3
2	2	4

このとき，*T* に対して，削除操作 ***d*** が発行されて，*T* は T_{new} になったとする．

d：DELETE FROM *T* WHERE *C*=4;

T_{new}

A	*B*	*C*
1	2	3
1	2	3
2	2	3

さて，検証するべき問題は，*T* は *R* と *S* の自然結合ビューであると定義されたので，もし *T* に対して発行された削除操作 ***d*** がローカルで削除可能なのであれば，体現化した結果が T_{new} となるような R_{new} と S_{new} が存在するべきである．そして，もし存在するとすれば，そのような R_{new} と S_{new} は R_{new}={(1, 2)(x_1), (2, 2)(x_2)}, S_{new}={(2, 3)(x_3) }と表されるであろう．そうすると，R_{new}*$_{bag}$$S_{new}$={(1, 2, 3)($x_1$×$x_3$), (2, 2, 3)($x_2$×$x_3$)}であり，一方，$T_{new}$ ={(1, 2, 3)(2), (2, 2, 3)(1)}であるから，次が成立しなければならない．

$x_1 \times x_3 = 2$ (1)

$x_2 \times x_3 = 1$ (2)

更に，次の問合せを *R* と *S* に発行することにより，各変数 x_1, x_2, x_3 の値の上限を知ることが出来る．

SELECT COUNT(*) AS x_1 FROM *R*
WHERE *A*=1 AND *B*=2;
SELECT COUNT(*) AS x_2 FROM *R*
WHERE *A*=2 AND *B*=2;
SELECT COUNT(*) AS x_3 FROM *S*
WHERE *B*=2 AND *C*=3;

この例では，x_1=2, x_2=1, x_3=1 となる．***d*** は削除操作であり，バッグでの重複元の識別性不可能原理により，x_1, x_2, x_3 の境界条件として次を得る．

x_1=0, 2，x_2=0, 1，x_3=0, 1 (0)

つまり，*T*に対する削除操作 ***d*** がローカルで受理可能かどうかは，(0), (1), (2)からなる非線形連立方程式が解をもつか否かで決定できる[5, 6].

そこで,この問題を整合ラベリング問題(CLP)として定式化し，それを併合法で解いてみる．まず，問題をCLP として形式化すると次のようである．

【定義 6】(CLP の定義) 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．

例題 5 に照らし合わせれば，これらは具体的に次のようである：

U={x_1, x_2, x_3}

L={0, 1, 2}

T={t_1, t_2}，ここに t_1= (x_1, x_3), t_2=(x_2, x_3),を表す．

R={R_1, R_2}，ここに R_1, R_2 はそれぞれ次の通りである：

R_1={(2, 1)}，R_2={(1, 1)}

ここに，2 項関係(t_i, R_i)を制約対，V_i=(t_i, R_i)を頂点という．この例では 2 つの頂点 V_1 と V_2 が定義され，その 2 頂点が併合され，一個の頂点 $V_{1,2}$ に縮退される．本稿では，その結果のみを記せば，$V_{1,2}$=({x_1, x_2, x_3}, {(2,1,1)})となる．つまり，x_1=2，x_2=1，x_3=1 であるべきであると，この場合は唯一の解が見つかった．従って，体現化した自然結合ビューが T_{new} となるような R_{new} と S_{new} は唯一存在し得て，従って *T*に対するローカルな削除操作 ***d*** は受理可能と結論できる(この場合，T は ***d*** の下で更新可能である)．ちなみに，R_{new}={(1, 2)(2), (2, 2)(1)}, S_{new}={(2, 3)(1) }である．換言すると，T_{new} では BMVD $B \Rightarrow\Rightarrow A|C$ が存在したということである．

5.4 ローカルな更新可能性の検証

本節では，マテリアライズドビューがローカルで更新可能か否かを検証した結果，受理できない削除操作の例を示す．

【例題 6】実テーブルの *supply* と *demand* は下に示した通りとし，体現化された自然結合ビュー*sd* を生成する．

supply

supplier	*part*
s1	p1
s1	p2
s2	p2

demand

part	*demander*
p1	d1
p2	d1
p2	d2

sd

supplier	*part*	*demander*
s1	p1	d1
s1	p2	d1
s1	p2	d2
s2	p2	d1
s2	p2	d2

sd に削除操作 ***d'***が発行されたとする．

d' : DELETE FROM *sd* WHERE
supplier=s2 AND *part*=p2 AND *demander*=d2;

d'の実行結果は次の通りとなる．

sd_{new}

supplier	*part*	*demander*
s1	p1	d1
s1	p2	d1
s1	p2	d2
s2	p2	d1

さて，***d'***はローカルで更新可能な削除操作であろうか？もしそうだとすれば，BMVD *part* ⇒⇒ *supplier*|*demander* が存在しないといけない．これを，CLP として形式化して解いてみると次のようである．もし，その BMVD が成立するのであれば，$supply_{new}$={(s1, p1)(x_1), (s1, p2)(x_2),(s2, p2)(x_3)}, $demand_{new}$={(p1, d1)(x_4), (p2, d1)(x_5), (p2, d2)(x_6)}が存在し，$x_1 \times x_4=1$, $x_1 \times x_5=0$, $x_1 \times x_6=0$, $x_2 \times x_4=0$, $x_2 \times x_5=1$, $x_2 \times x_6=1$, $x_3 \times x_4=0$, $x_3 \times x_5=0$, $x_3 \times x_6=1$, $x_1=0, 1$, $x_2=0, 1$, $x_3=0, 1$, $x_4=0, 1$, $x_5=0, 1$, $x_6=0, 1$ である．従って，$t_1=(x_1, x_4)$, $t_2=(x_1, x_5)$, $t_3=(x_1, x_6)$, $t_4=(x_2, x_4)$, $t_5=(x_2, x_5)$, $t_6=(x_2, x_6)$, $t_7=(x_3, x_4)$, $t_8=(x_3, x_5)$, $t_9=(x_3, x_6)$,そして，R_1={(1, 1)}, R_2={(0, 0), (0, 1), (1, 0)}, R_3={(0, 0), (0, 1), (1, 0)}, R_4={(0, 0), (0, 1), (1, 0)}, R_5={(1, 1)}, R_6={(1, 1)}, R_7={(0, 0), (0, 1), (1, 0)}, R_8={(0, 0), (0, 1), (1, 0)}, R_9={(1, 1)}である．この場合，$V_{1,2,3,4,5,6,7,8,9}$=({ x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9},φ)，ここにφは空集合を表す，となる．つまり，解はなく，従って，***d'***を受理してはいけない．即ち，***d'***はローカルで受理可能な削除操作ではない．

5.5 SQL によるローカルな更新可能性の検証

併合法を用いた整合ラベリング問題は SQL の自然結合問合せとして書き下すことが出来る．例えば，例題 5 の場合，$R_{1,3}(x_1, x_3)$={(2, 1)}, $R_{2,3}(x_2, x_3)$={(1, 1)}という 2 つのリレーションが定義され，この 2 つのリレーションの自然結合をとると，$R_{1,2,3}=R_{1,3}*R_{2,3}$={(2, 1, 1)}となる．つまり，$x_1=2$, $x_2=1$, $x_3=1$ が唯一の解であることを示しており，例題 5 でこの問題を非線形連立方程式を併合法で解いたのと同じ解が得られていることが分かる．

一方，例題 6 の場合，$R_{1,4}(x_1, x_4)$={(1, 1)}, $R_{1,5}(x_1, x_5)$={(0, 0), (0, 1), (1, 0)}, $R_{1,6}(x_1, x_6)$={(0, 0), (0, 1), (1, 0)}, $R_{2,4}(x_2, x_4)$={(0, 0), (0, 1), (1, 0)}, $R_{2,5}(x_2, x_5)$={(1, 1)}, $R_{2,6}(x_2, x_6)$={(1, 1)}, $R_{3,4}(x_3, x_4)$={(0, 0), (0, 1), (1, 0)}, $R_{3,5}(x_3, x_5)$={(0, 0), (0, 1), (1, 0)}, $R_{3,6}(x_3, x_6)$={(1, 1)} であり，$R_{1,2,3,4,5,6} = R_{1,4}*R_{1,5}*R_{1,6}*R_{2,4}*R_{2,5}*R_{2,6}*R_{3,4}*R_{3,5}*R_{3,6}=$φ となるので，例題 6 で示したように，この問題の解はないことが分かる．

6 おわりに

マテリアライズドビューの更新問題を論じ，ローカルで更新可能なマテリアライズドビューという新概念を導入し，それがバッグ MVD という概念で定式化できることを示した．更に，この問題を整合ラベリング問題として定式化することが出来て，それを併合法を用いて解くことで，更新可能なマテリアライズドビューやローカルで更新可能なマテリアライズドビューを規定できることを示した．

なお，更新可能なマテリアライズドビューの場合の実テーブルの更新メカニズムの実装法やローカルで更新可能なマテリアライズドビューサポートの実装法は今後の課題である．

【謝辞】本研究を遂行するにあたり，マテリアライズドビューとその更新に関して，SQL 標準での取り扱われ方について的確に解説して下さった小寺孝氏(日立製作所)に感謝する．

参 考 文 献

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

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

[3] Grumbach, S. and Milo, T., Towards Tractable Algebras for Bags, *J. of Computer and System Sciences*, 52(3), pp.570-588, 1996.

[4] Gupta, A. and Mumick, I. S. (Eds), Materialized Views – Techniques, Implementations, and Applications, The MIT Press, Cambridge, Massachusetts, 1998.

[5] 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.

[6] Masunaga, Y., Nagata, Y. and Ishii, T., Making Join Views Updatable on Relational Database Systems in Theory and in Practice, *Proc. IMCOM2019*, 2019.

[7] 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.

[8] Oracle, マテリアライズド・ビューを使用した読取り/書込みデータのレプリケート https://docs.oracle.com/cd/E16338_01/server.112/b56305/tdpii_reppit.htm#BEHCEFGC

[9] Oracle, 読取り専用、更新可能および書込み可能のマテリアライズド・ビュー https://docs.oracle.com/cd/E16338_01/server.112/b72089/repmview.htm#BABCCHDA

[10] Oracle, アドバンスト・レプリケーションのサポート終了 https://docs.oracle.com/cd/E82638_01/upgrd/desupported-features-oracle-database-12c-r2.html#GUID-32AC17C8-B9CB-44D5-B6B6-CB8D1AA8E307

[11] Tsang, E., Foundations of Constraint Satisfaction, Herstellung und Verlag: BoD, 1996.