Vol.2018-DBS-167 No.16
Vol.2018-IFAT-132 No.16
2018/9/13

OID を用いた新規 Incremental View Maintenance 方式の提案

増永良文[†1]　長田悠吾[†2]　石井達夫[†2]

概要：体現化ビュー（materialized view）はデータベースに格納された実体をもつビューなので，それを定義するために使われている実テーブルに更新があると，その更新を反映するべく体現化ビューも再構築されねばならない．しかしながら，そのための再構築を一から行うのではコストが掛かるので奨められず，実テーブルへの変化分に応じた更新だけを実体化ビューに施すという IVM（incremental view maintenance）が研究され，そのためのアルゴリズムが提案されている．これまで，タップルに基づくアプローチやそれより効率的であるとする ID に基づくアプローチが提案されてきた．しかし，ID に基づくアプローチは（集合意味論ではなく）バッグ意味論に立脚している SQL が定義する体現化ビューでは機能しない．本報告では，OID（object identifier，オブジェクト識別子）を利用することにより，バッグ意味論の下でも機能する新規 IVM 方式を提案している．

キーワード：体現化ビュー，マテリアライズドビュー，漸増ビューメンテナンス，IVM，OID，SQL，バッグ意味論，リレーショナルデータベース，データウェアハウス，OLAP，ビジネスインテリジェンス

Proposal of a Novel Incremental View Maintenance Mechanism Using OIDs

YOSHIFUMI MASUNAGA[†1]　YUGO NAGATA[†2]　TATSUO ISHII[†2]

1. はじめに

体現化ビュー (materialized view，マテリアライズドビュー)はビジネスインテリジェンス分野における OLAP のためのデータウェアハウスでよく用いられている．しかしながら，体現化ビューをメンテナンスしていくためには，体現化ビューを定義している実テーブルが更新されると，それに伴い体現化ビューも更新されねばならない．しかしながら，体現化ビューを一から再構築するのはコストが掛かるので，それに代わる効率的な方法はないのかが問題となる．それが IVM（incremental view maintenance，漸増ビューメンテナンス）である．これまで多くの IVM 研究・開発が報告されてきているが[1, 2]，これまでのタップルアプローチ（tuple-based approach）に代わって，最近 ID アプローチ（ID-based approach）が提案され，それがタップルアプローチと比較してより効率的であると報告されている[3]．しかしながら，ID アプローチは，実テーブルにキーが存在し，かつ体現化ビューにそのキーが継承されねばならないという制約があるため，テーブルに重複した行があってもよいとする「バッグ意味論」のもとでは機能しないという欠点を有する．本研究はこの問題を解決するために，リレーショナル DBMS がテーブルの行を生成する際に付与している OID（object identifier、オブジェクト識別子）に着目することにより, バッグ意味論の下でも IVM を実現できる新しいアプローチ，これを「OID アプローチ」（OID-based approach）という，を提案する．

†1 お茶の水女子大学（名誉教授）
Ochanomizu University (Professor Emeritus)
†2 SRA OSS 日本支社
SRA OSS, Inc. Japan

2. Incremental View Maintenance とは何か

2.1 体現化ビューの定義

バッグ代数（bag algebra, BA）は単項演算（重複除去，選択，射影），2 項演算（和，差，共通，直積，結合），グループ化演算からなる[4, 5]．バッグ代数はバッグ意味論のもとでの問合せやビュー定義を指定するために使われる．このとき，バッグ代数表現は（一般にはバッグである）実テーブル群に対して，これらの演算を再帰的に適用して定義される．ここに，P は選択条件，A は属性集合，φは結合条件，$\gamma_{G, f(M)\rightarrow g}(V)$は次に示すグループ化演算を表す．

SELECT G, f(X) AS g FROM V
WHERE ・・・ GROUP BY G

このとき，(Q_v, T_v)で体現化ビューを表す．ここに，Q_v はビューを定義する問合せ（＝BA 表現），T_v はこの時点で体現化されたビューのコンテンツを表す．従って，D をその時点でのデータベースの状態とすれば, $T_v=Q_v(D)$である．添え字の v はビューを意味する．

2.2 IVM の定義

体現化ビューは実テーブルに更新があった場合，その体現化ビューをメンテナンスする原始的な方法は，更新された実テーブルを使って，改めて一からビューを再構築することであるが，コスト（＝処理のためにアクセスするページ数や CPU コスト）がかかり，工夫のない方法である．そのために，更新により実テーブルに施された変化分（＝削除される行や追加される行）に着目して，何とかその変化分に対応して，データベースに格納されている体現化ビューの一部だけを更新することで，体現化ビューのメンテナンスを行うことができないかと考える．この処方が，IVM

Vol.2018-DBS-167 No.16
Vol.2018-IFAT-132 No.16
2018/9/13

（incremental view maintenance）である．

(Q_v, T_v)を体現化ビューとするとき，IVM の定義は図 1 に示す通りである[2]. データベース D に更新δD が掛ったとき，$Q_v(D')$を再計算により求めるのではなく，T_v に対する変化分δT_v を如何にして求めるか，それを問題にするのが IVM である．

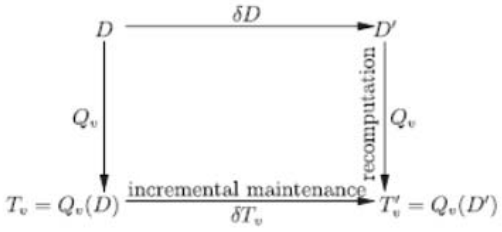


図 1　IVM の定義

Figure 1　The definition of IVM.

2.3 タップルアプローチ　vs　ID アプローチ

タップルアプローチと ID アプローチを比較する[3].

【例題 1】2 つの実テーブル製品と部品，そして体現化ビューV を次のように定義する．ここに，製品の主キーは{製品名, 部品番号}，部品の主キーは{部品番号}である．

製品

製品名	部品番号
G1	P1
G2	P1
G3	P2

部品

部品番号	価格
P1	10
P2	20

```
CREATE MATERIALIZED VIEW V AS
SELECT 製品名，部品番号，価格
FROM 製品 NATURAL JOIN 部品
```

V

製品名	部品番号	価格
G1	P1	10
G2	P1	10
G3	P2	20

さて，このとき，実テーブル商品の部品 P1 の価格が 10 から 11 に更新されたとしよう. この更新 u による実テーブル商品に対する差分テーブルは次のようになる．

δ^u(部品)

部品番号	旧価格	新価格
P1	10	11

両アプローチの大きな違いは，この差分テーブルをどのように体現化ビューV に反映するかである．

- **タップルアプローチの場合の V の差分**

体現化ビューのどのタップルをどう更新しないといけないか，を指示するための差分（diffs）は次のテーブルで表される．なぜならば，体現化ビューV の主キーが{顧客名，商品番号}であるからである．

δ^u(V)

製品名	部品番号	旧価格	新価格
G1	P1	10	11
G2	P1	10	11

- **ID アプローチの場合の V の差分**

一方，ID アプローチの場合，更新のあった実テーブル商品の主キーが体現化ビューV にそのまま伝搬しているので，V に対して発行されるべき差分は次のテーブルでよい．

δ^u(V)

部品番号	旧価格	新価格
P1	10	11

□

2.4 ID アプローチの問題点

さて, ID アプローチはビューに実テーブルのキーが伝搬されていることが条件となっている．そのキーを頼りに IVM を行っている．このことは，裏を返せば，もし更新のあった実テーブルのキーが体現化ビューに伝搬していなければ ID アプローチは機能しない．このことを，例題 2 で確認すると共に，では如何にしてこの問題点を解決していけばよいのか，論じていく．

【例題 2】実テーブル納品を次のようにする．ここに，納品テーブルの主キーは{顧客名, 製品番号}である．ただし，商品番号→商品名ではあるが，商品番号→単価ではない．つまり, 大口顧客には単価を安くしたいという想定である．

納品

顧客名	商品番号	商品名	単価
C1	I1	テレビ	20
C1	I2	冷蔵庫	30
C2	I1	テレビ	22
C2	I2	冷蔵庫	30

このとき，体現化ビュー　卸値＝$\pi_{\{商品番号,\ 単価\}}$(納品) を次のように定義する．

```
CREATE MATERIALIZED VIEW 卸値(商品番号，単価)
AS　SELECT 商品番号，単価 FROM 納品
```

その結果, 次のような体現化ビューができあがる. 我々は，バッグ意味論の下で IVM を論じているので, 重複したタップルはそのまま表れる．従って，キーもない．

卸値

商品番号	単価
I1	20
I2	30
I1	22
I2	30

Vol.2018-DBS-167 No.16
Vol.2018-IFAT-132 No.16
2018/9/13

さて，C1 は大口顧客なので，商品番号 I2 の卸値を 30 から 29 に引き下げたく，更新要求 w を発行したとしよう．

w: UPDATE 納品 SET 単価＝29
WHERE 顧客名＝‘ C1 ’ AND 商品番号＝‘ I2 ’

このとき，以下のことが観察される．

- **ID アプローチの場合**

ビュー卸値には実テーブル納品のキーが残存しないので，ID アプローチは適用不可能である．

- **タップルアプローチの場合**

実テーブル納品に対して，w という更新がなされたので，納品テーブルに対する差分 $\delta^{w}_{納品}$は次のとおりである．

$\delta^{w}_{納品}$

顧客名	商品番号	商品名	単価旧値	単価新値
C1	I2	冷蔵庫	30	29

このとき，ビュー卸値は納品テーブルの{商品番号，単価}上の射影なので，タップル差分（diff）の変換ルールにより，$\delta^{w}_{納品}$は次のようなビュー卸値の diff に変換される．しかし，$\delta^{w}_{卸値}$によりビュー卸値の行(I2, 30)が 2 本とも(I2, 29)に更新されてしまい，誤った IVM となる．

$\delta^{w}_{卸値}$

商品番号	単価旧値	単価新値
I2	30	29

□

つまり，これまでのアプローチはいづれも機能しない．

3. OID アプローチ

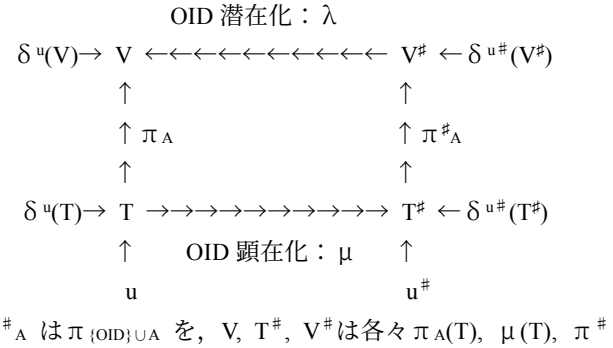
3.1 OID とは

OID（object identifier，オブジェクト識別子）は商用あるいは OSS のリレーショナル DBMS，たとえば Oracle database や PostgreSQL で実装されている．OID 付与の環境下でテーブルを生成しておくと[a]，そのテーブルに新しく行が挿入されると，その行には DBMS により OID が自動的に付与される．OID は PostgreSQL では 32 ビットの整数である．なお，行を識別するためならば，OID ではなく行 ID（row ID）を使えばよいと考えるかもしれないが，行 ID はデータベースの再構築で一般に変化してしまうが，OID は不変である．

3.2 OID アプローチのしくみ

実テーブル T(A, ..., B)を WITH OIDS オプションで生成するとする．このとき，μ(T)で T(OID, A, ..., B)）を表すこととする．操作μを OID 顕在化（manifestation）という．一方，μ(T)を T に戻す操作λを OID 潜在化（latentization）ということにする．つまり，λ(μ(T)) = T である．

さて，OID アプローチのしくみを射影ビューを例にして，図 2 に示す．T に更新 u が施された場合に生じる差異δ^{u}(T)からビューV に対する差異δ^{u}(V)をどのように求めるかが問われたが，従来のアプローチは，図 2 の左側の T から V に至るπ_Aと名付けられたパスで差異の伝搬を計算していた．一方，本アプローチは，図 2 の右側に示すとおり，T に一旦 OID 顕在化を施して，OID のある世界で差異δの計算を行う．得られた結果に対して OID 潜在化を施すことにより，V に対する所望の差異を求める．



$\pi^{\sharp}_{A}$ は$\pi_{\{OID\}\cup A}$ を，V, $T^{\sharp}$, $V^{\sharp}$は各々π_A(T), μ(T), $\pi^{\sharp}_{A}$($T^{\sharp}$)を表す．δ^{u}(T)は更新 u による T の差異（δ-スクリプト）を表す（V, $T^{\sharp}$, $V^{\sharp}$に対しても同様）．

図 2 OID アプローチのしくみ―射影演算の場合―

Figure 2 Schematic OID-based approach: Projection operation

3.3 OID に基づく IVM 方式

実テーブルの更新によって生じる差異δがどのように体現化ビューに伝搬され，結果としてビューの IVM が可能となるのか，その全体像を明らかにしておく必要がある．それを IVM 方式という．OID アプローチが体現化ビュー定義の演算として取り扱うのは，バッグ意味論の下での選択，射影，重複除去，和，差，共通，直積，結合，グループ化であるので，個々のビュー定義演算に対して，差分δの伝搬可能性を論じなければならない．

3.4 OID アプローチでの差異の伝搬

体現化ビューを定義するために使われる演算の対象となったテーブルを定義域テーブル（domain table），演算を施された結果出来上がったテーブルを値域テーブル（range table）ということにすれば，定義域テーブルに対する更新がもたらした差異とは，新旧定義域テーブルの差であり，差異の伝搬とはその差異を的確に値域テーブルに伝えて，値域テーブルを IVM することをいう． 2 点注意する．

（1）先述したように WITH OIDS オプションで生成した実テーブルの行には OID が付与されるが，導出表には OID は付かない．提案している OID アプローチでは際の伝搬に当たり，行の唯一識別性を保証してくれる列が常に必要で，その目的のために OID_GEN と名付けた単項のテーブルを WITH OIDS オプション付きで生成する．

（2）OID 顕在化を施される前のテーブル T は一般的にバッグであって構わないが，OID が顕在化した $T^{\sharp}$に重複した行はなく集合（＝リレーション）となっている．従って，OID 顕在化の世界では，基本的にはリレーショナル代数が適用可能となる．しかしながら，OID 顕在化前のテーブル T が一般にはバッグであることを受けて，$T^{\sharp}$に対するリレ

a 例えば PostgreSQL では，create table 時に with oids オプションを付与する．

ーショナル代数演算や更新操作には適宜変更を余儀なくされるところがある．例えば，$T^{\#}$={(oid1, t), (oid2, t)}から(oid1, t)だけを選択したり，削除することは許されない．

(1) 選択ビューと差異の伝搬

【例題 3】T(A, B) = {(1, 1), (1, 1), (1, 2), (2, 1), (2,2)}とし，Tの選択体現化ビューV を次のとおり定義する．

CREATE MATERIALIZED VIEW V AS
SELRCT * FROM T WHERE B=1

バッグ代数表現では，$V=\sigma_{B=1}(T)$であり，V={(1, 1), (1, 1), (2, 1)}である．T に次に示す更新 u が発行されたとしよう．

u: UPDATE T SET B=B+1 WHERE A=2

さて，$T^{\#}$={(1000, 1, 1), (1001, 1, 1), (1002, 1, 2), (1003, 2, 1), (1004, 2,2)}としよう．u は $T^{\#}$への更新 $u^{\#}$に機械的に変換される．

$u^{\#}$: UPDATE $T^{\#}$ SET B=B+1 WHERE A=2

このとき，$T^{\#}$の差異$\delta_{u^{\#}}(T^{\#})$が求められる．

$\delta_{u^{\#}}(T^{\#})$

OID	B 旧値	B 新値
1003	1	2
1004	2	3

ところで，$V^{\#}=\sigma^{\#}_{B=1}(T^{\#})$= {(1000, 1, 1), (1001, 1, 1), (1003, 2, 1) }である．この例では，u が更新の対象とした行は A 値が 2 の行，つまり OID が 1003 と 1004 の行であったが，ビューV は B 値が 1 の行，つまり OID が 1000, 1001, 1003 の行だったので，結果として$\delta_{u^{\#}}(T^{\#})$は $V^{\#}$への差異$\delta_{u^{\#}}(V^{\#})$に，次に示されるように変換される．

$\delta_{u^{\#}}(V^{\#})$

OID	B 旧値	B 新値
1003	1	2

(2) 射影ビューと差異の伝搬

【例題 4】納品テーブル次のように生成する．

CREATE TABLE 納品(顧客名 NARCHAR(10), 商品番号 CHAR(10), 商品名 NVARCHAR(20), 単価 INT) WITH OIDS

この納品テーブルに対して，行を適当に挿入して，

SELECT OID, 納品.* INTO 納品$^{\#}$
FROM 納品

を実行すると，下記の導出表 μ(納品)，これが納品$^{\#}$，が得られたとする．

納品$^{\#}$

OID	顧客名	商品番号	商品名	単価
1000	C1	I1	テレビ	20
1001	C1	I2	冷蔵庫	30
1002	C2	I1	テレビ	22
1003	C2	I2	冷蔵庫	30

一方，体現化ビュー卸値を次のように定義する．

CREATE MATERIALIZED VIEW 卸値(商品番号，単価)
AS SELECT 商品番号, 単価 FROM 納品

この卸値ビューは次のとおりである．

卸値

商品番号	単価
I1	20
I2	30
I1	22
I2	30

注意するべき点は，卸値を体現化ビューと定義したので，データベースには格納されるが，その行に OID は付与されないことである．従って，卸値そのままでは OID アプローチは適用できない．つまり，DBMS が実テーブルの行に付与した OID をいかにしてビューを構成する行に伝搬するかが問われることになる．

そこで，実テーブル納品に対して，例題 2 と同じ更新要求 w が発行されたとする．OID アプローチでは，図 2 の右側に示される $T^{\#}$から $V^{\#}$に至るパスを論じることになるので，まず w を $w^{\sharp}$に，納品を納品$^{\#}$に置き換えて機械的に問合せとテーブルを改変する．

$w^{\#}$: UPDATE 納品$^{\#}$ SET 単価＝29
WHERE 顧客名='C1' AND 商品番号='I2'

$w^{\#}$を納品$^{\#}$に施すと，差異$\delta_{w^{\sharp}}$(納品$^{\#}$)が求められる．

$\delta_{w^{\sharp}}$(納品$^{\#}$)

OID	単価旧値	単価新値
1001	30	29

なお，この表現は次のトランザクションを実行して，納品$^{\#}$の新旧を比較して得られる．

BEGIN TRANSACTION;
DELETE FROM 納品$^{\#}$ WHERE 顧客名='C1' AND 商品番号='I2';
INSERT INTO 納品$^{\#}$ VALUES (1001, C1, I2, 冷蔵庫, 29);
END;

さて，次に問われるのは，差異$\delta_{w^{\sharp}}$(納品$^{\#}$)が射影演算$\pi^{\#}_{\{商品番号,\ 単価\}}$の下で，どのように $V^{\#}=\pi^{\#}_{\{商品番号,\ 単価\}}$(納品$^{\#}$)への差異に変換されうるかである．この例では，w が更新の対象とした列である単価が射影されているので，$\delta_{w^{\sharp}}$(納品$^{\#}$)は $V^{\#}$への差異$\delta_{w^{\sharp}}(V^{\#})$に変換できる．

$\delta_{w^{\sharp}}(V^{\sharp})$

OID	単価旧値	単価新値
1001	30	29

一方，$V^{\#} = \pi^{\#}_{\{商品番号,\ 単価\}}$ (納品$^{\sharp}$)は次のようであった．

$V^{\#}$

OID	商品番号	単価
1000	I1	20
1001	I2	30
1002	I1	22
1003	I2	30

従って，$V^{\#}$に$\delta^{w\sharp}(V^{\sharp})$を施すと，$V^{\#}$は更新され$V^{\#}{}_{new}$となる．

$V^{\#}{}_{new}$

OID	商品番号	単価
1000	I1	20
1001	I2	29
1002	I1	22
1003	I2	30

$V^{\#}{}_{new}$に潜在化λを施すと，OID 列が削除されて，新たなビュー卸値 $_{new}$ が得られるが，これはまさしく，実テーブル納品へ更新要求 w がかけられ，その結果を反映したビューの更新結果である．

卸値 $_{new}$

商品番号	単価
I1	20
I2	29
I1	22
I2	30

□

この例題は，これまでのタップルアプローチや ID アプローチでは扱えなかったが，このように，OID アプローチを採用すると射影ビューの IVM が実現できている．

（3） 重複除去ビューと差異の伝搬

$T^{\#}$は集合だから重複した行はないが，T ではあり得るので，$\varepsilon(T)$，つまり，SELECT DISTINCT * FROM T　が発行された場合に，$\varepsilon^{\#}(T^{\#})$をどうするかが問われる．そのためには，$T^{\#}$で OID 値以外はすべて同じ値を持つ行の集まりに$\varepsilon^{\#}$演算を施した結果が，唯一識別可能な OID をもつ 1 行になってほしい．そのために，この方式では，（新規 OID 生成専用に）OID_GEN と名付けた単項のテーブルを WITH OIDS オプション付きで生成し，そこで必要に応じて行を 1 本生成し，その行に付与された OID を頂いて，その他の値はすべて重複削除の対象となった行の値とする行を作成して，それを伝搬結果とする．なお，差異の伝搬は，葉から根（＝ビュー）への一方向なので，旧 OID の集合と新規 OID との対応関係は，その意味では保存不要である．

（4） 和ビューと差異の伝搬

バッグ意味論では，和には加法和と最大和があるb)が，集合意味論での和は加法的なので，両立性の観点から和は加法和とする．これは，SQL の重複行をそのまま扱う UNION ALL に相当する．R と S を和両立とすると，和ビューは $R \cup S = \{t_i(k_i) \mid t_i \in \varepsilon(R) - \varepsilon(S)\} \cup \{t_i(k_i + \ell_j) \mid t_i = u_j \in \varepsilon(R) \cap \varepsilon(S)\} \cup \{u_j(\ell_j) \mid u_j \in \varepsilon(S) - \varepsilon(R)\}$，ここに，$t_i = u_j \in \varepsilon(R) \cap \varepsilon(S)$は $t_i \in \varepsilon(R) \wedge u_j \in \varepsilon(S) \wedge t_i = u_j$ の短縮形である．

そこで，和 $V^{\#} = (R \cup S)^{\#}$を考える．$R^{\#}, S^{\#}$は共に集合であり，もし $R^{\#} \cap S^{\#} = \phi$ ならば，$V^{\#} = R^{\#} \cup {}^{\#}S^{\#}$となる．もし，$R^{\#} \cap S^{\#} \neq \phi$の場合，$(OID, a, ..., b) \in R^{\#} \cap S^{\#}$とするとき，OID_GEN を使って OID を新規に 2 個生成し，これらを OID_{new1} と OID_{new2} とする，$(OID_{new1}, a, ..., b)$と$(OID_{new2}, a, ..., b)$を作り，それらを $V^{\#}$の元とする．なお，$R^{\#}$や $S^{\#}$は集合なので，共通する行はあったとすれば 1 本同士でしかない．このような議論が必要なのは，バッグ意味論の下では $R \cup R \neq R$ だからである．

（5） 差ビューと差異の伝搬

R と S を和両立とし，バッグ意味論での R と S の差（monus）R－S の定義を確認しておく．$R - S = \{t_i(k_i) \mid t_i \in \varepsilon(R) - \varepsilon(S)\} \cup \{t_i(\max(0, (k_i - \ell_j))) \mid t_i = u_j \in \varepsilon(R) \cap \varepsilon(S)\}$．これは，SQL の重複行をそのまま扱う EXCEPT ALL に相当する．差異の伝搬は，一般に，R に更新が掛かった場合，S に更新が掛かった場合，そして両者に更新が掛かった場合に分けられるが，詳細は紙面の都合上，省略する．

（6） 共通ビューと差異の伝搬

R と S を和両立とし，バッグ意味論での R と S の共通 $R \cap S$ の定義は，$R \cap S = \{t_i(\min(k_i, \ell_j)) \mid t_i = u_j \in \varepsilon(R) \cap \varepsilon(S)\}$である．これは，SQL の重複行をそのまま扱う INTERSECT ALL に相当する．次に $V^{\#} = (R \cap S)^{\#}$を考える．例で説明する．

【例題 5】$R(A) = \{1(1), 2(2), 3(2)\}$, $S(A) = \{1(2), 2(2), 3(1)\}$とする．このとき，$R \cap S = \{1(1), 2(2), 3(1)\}$である．定義から，$R \cap S$ の元は R の元でもあり S の元でもあるということは分かるが，バッグ意味論の下では，どちらのテーブルのどの元（＝行）とは特定できない．そこで，OID_GEN に OID を生成してもらい，$V^{\#}$の各行に付与することとする．

このとき，一般性を失うことなく，R に例えば次のような更新要求 u が発行されたとする．

u: INSERT INTO R VALUES 1

$R^{\#}$の差異は次のとおりである．

$\delta^{u\sharp}(R^{\#})$

OID	A 旧値	A 新値
1010	NULL	1

この差異は $V^{\#}$へ伝搬されて，そこでの差異となる．

$\delta^{u\sharp}(V^{\#})$

OID	A 旧値	A 新値
3004	NULL	1

（7） 直積ビューと差異の伝搬

b {a,b,b}と{a,a,b}の加法和は{a,a,a,b,b,b}，最大和は{a,a,b,b}である．

まず，R と S の直積 R×S の OID 表現を考える．ここで，一般性を失うことなく，R と S は WITH OIDS オプションの指定のもと生成されたとする．そうすると，R と S に OID 顕在化を施して，$R^\sharp$と $S^\sharp$が得られたとする．このとき，$R^\sharp \times S^\sharp$の OID 付与を考える．$R^\sharp \times S^\sharp = \{(OID_R, a, ..., b, OID_S, c, ..., d) \mid (OID_i, a, ..., b) \in R^\sharp \wedge (OID_j, c, ..., d) \in S^\sharp\}$である．そこで，各$(OID_i, a, ..., b, OID_j, c, ..., d)$に対して，OID_GEN から新規 OID_{ij}を生成してもらい，$(R \times S)^\sharp \equiv \{(OID_{ij}, a, ..., b, c, ..., d) \mid (OID_i, a, ..., b, OID_j, c, ..., d) \in R^\sharp \times S^\sharp\}$と定義する．

【例題 6】R(A)={1, 2, 2}，S(B)={1, 1, 2}とし，直積ビュー V=R×S の R に更新 u が掛かったとする．

u: UPDATE R SET A=A+1 WHERE A=2

一般性を失うことなく，$R^\sharp$={(1000, 1), (1001, 2), (1002, 2)}, $S^\sharp$={(1003, 1), (1004, 1), (1005, 2)}とする．このとき，$u^\sharp$による $R^\sharp$の差異$\delta_{u^\sharp}(R^\#)$は次のように計算される．

$\delta_{u^\sharp}(R^\#)$

OID	A 旧値	A 新値
1001	2	3
1002	2	3

さて，$V^\# = (R \times S)^\sharp$は 9 本の行からなるが，そのうちの 6 本が$\delta_{u^\sharp}(R^\#)$の影響を受ける．そして，その差異は次のように書ける．ここに，OID は OID_GEN が生成したものである．たとえば，OID の 3000 は OID の組(1000, 1003)に対して OID_GEN が生成した新規 OID かもしれない．

$\delta_{u^\sharp}(V^\#)$

OID	A 旧値	A 新値
3000	2	3
・・・・・・		
3005	2	3

（8） 結合ビューと差異の伝搬

テーブル R と S の結合にはθ‐結合，自然結合，内結合，外結合，準結合演算などがある．演算の独立性の観点からは，いづれも直積，選択，射影演算を適当に組み合わせて表現できる．従って，結合ビューへの差異の伝搬はそれらの演算を順次適用して得られることになる．本報告では紙面の都合上，詳細を割愛することとする.

（9） グループ化ビューと差異の伝搬

T にグループ化演算が発行され，体現化ビューV が定義されたとする．

CREATE MATERIALIZED VIEW V AS

SELECT G, f(M) AS E FROM T WHERE c GROUP BY G

グループ化列 G の異なる値ごとに M を引数とする集約関数 f が実行されて，それが E 値として，2 項の導出表が得られる．$R^\#$に当該集約演算を施した結果は，R にグループ化を施して得られる導出表の各行に，OID_GEN で生成した OID をそれぞれ付与して得られる．

さて，T に更新要求 u が発行されたとする．これは $T^\#$に $u^\sharp$が発行されたということになるが，ではその差分 $\delta(u^\sharp(T^\#))$は $V^\#$にどのように伝搬されるのか，例に見る．

【例題 7】T(A, B) = {(1, 1)(2), (1, 2)(1), (2, 1)(1), (2,2)(1)}

CREATE MATERIALIZED VIEW V AS

SELECT A, sum(B) AS C FROM T GROUP BY A

V(A, C)={(1, 4), (2, 3)}である．$V^\#$は OID_GEN で新規獲得した OID，たとえば 3000 と 3001 を付与して，$V^\#$= {(3000, 1, 4), (3001, 2, 3)}となる．ちなみに，$T^\#$= {(1000, 1, 1), (1001, 1, 1), (1002, 1, 2), (1003, 2, 1), (1004, 2,2)}とする．

そこで，T に対して更新要求 u が発行されたとする．

u: UPDATE T SET B=B+1 WHERE A=1 AND B=1

この要求は機械的に $T^\#$への要求 $u^\#$に変換される．このときの $u^\#$による $T^\#$の差異は次のとおりである．

$\delta_{u^\sharp}(T^\#)$

OID	A 旧値	A 新値
1000	1	2
1001	1	2

このとき，このグループ化演算による OID の対応関係は，{1000, 1001, 1002}→3000, {1003, 1004}→3001 であるので，$V^\#$の差異は次の通りとなる．

$\delta_{u^\sharp}(V^\#)$

OID	A 旧値	A 新値
3000	4	6

4. まとめと今後の課題

体現化ビューの IVM（incremental view maintenance）について，従来のタップルアプローチや ID アプローチでは IVM が機能しない体現化ビューでも IVM が可能となる OID アプローチを提案した．

謝辞 本研究は JSPS 科研費 16K00152 の助成を受けている．

参考文献

[1] Ashish Gupta and Inderpal Singh Mumick (ed.), Materialized Views: Techicniques, Implementations, and Applications, The MIT Press, 589p., 1999.

[2] Rada Chirkova and Jun Yang, Materialized Views, Foundations and Trends in Databases 4(4), pp.295-405, 2012.

[3] Yannis Katsis, Kian Win Ong, Yannis Papakonstantinou and Kevin Keliang Zhao, Utilizing IDs to Accelerate Incremental View Maintenance, Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (SIGMOD '15), pp. 1985-2000, 2015.

[4] Timothy Griffin and Leonid Libkin, Incremental Maintenance of Views with Duplicates, Proc. ACM SIGMOD'95, pp.328-339, 1995.

[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, 8p. 2017.

正誤表

下記の箇所に誤りがございました．お詫びして訂正いたします．

訂正箇所	誤	正
2 ページ左欄 ↑6, 8 行目	商品	部品
2 ページ右欄 ↓4 行目	商品	部品
3 ページ右欄 ↑10 行目	際	差異
5 ページ右欄 ↓7～15 行目	$R^{\#}$, $S^{\#}$は共に集合であり，もし $R^{\#} \cap S^{\#} = \phi$ ならば，$V^{\#}=R^{\#} \cup {}^{\#}S^{\#}$となる．もし，$R^{\#} \cap S^{\#} \neq \phi$ の場合，(OID, a, ..., b)$\in R^{\#} \cap S^{\#}$とするとき，OID_GEN を使って OID を新規に 2 個生成し，これらを OID_{new1} と OID_{new2} とする，$(OID_{new1}, a, ..., b)$と$(OID_{new2}, a, ..., b)$を作り，それらを $V^{\#}$の元とする．なお，$R^{\#}$や $S^{\#}$は集合なので，共通する行はあったとすれば 1 本同士でしかない．このような議論が必要なのは，バッグ意味論の下では $R \cup R \neq R$ だからである．	もし $\varepsilon(R) \cap \varepsilon(S) = \phi$ ならば，$V^{\#}=R^{\#} \cup {}^{\#}S^{\#}$となる．もし，$\varepsilon(R) \cap \varepsilon(S) \neq \phi$ の場合，$(a, ..., b)(k) \in \varepsilon(R) \cap \varepsilon(S)$ ならば，OID_GEN を使って OID を新規に k 個生成し，これらを $OID_{new1}, ..., OID_{newk}$ とする，$(OID_{new1}, a, ..., b), ..., (OID_{newk}, a, ..., b)$を作り，それらを $V^{\#}$の元とする．