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

増永 良文 [1, a)]

概要：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にしか過ぎないので，ビューをさらに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”しようとするとその更新可能性が問題となってくる．このビュー更新問題はビューが導入された時点から，多くの研究者と実務家の関心を惹き，数多くの研究・開発がなされてきたが，未だその全貌が明らかにされずに今日に至っている．本報告ではリレーショナルデータベースのビュー更新可能性について，まず，和集合ビュー，差集合ビュー，共通集合ビュー，直積ビュー，射影ビュー，選択ビュー，結合ビューの 7 つの基本的なビューの更新可能性について，“更新意図の外形的推測に基づく更新可能性”という新しい考え方を導入することにより，これらすべてのビューの更新可能性を体系的に明らかにする．さらに，この結果を基にして，これら 7 つの演算を再帰的に用いて定義される一般的なビューの更新可能性について，その更新可能性を判定するアルゴリズムを示す．本報告の更新意図の外形的推測に基づくアプローチが従来型アプローチと根本的に異なる点は，後者で更新不可能とされる場合でも，ビュー更新の変換候補の“外形”を，ビューを一時的にマテリアライズして仮計算してみると，ユーザが発行した更新要求の意図を一意に“推測”できうる場合がある点に着目したところにある．その結果，これまで完全な解が与えられていなかったビュー更新問題に統一的な解を示し得たものと考えられる．

Updatability of Relational Database Views based on Pro Forma Guessing of Update Intention

YOSHIFUMI MASUNAGA[1, a)]

Abstract: Given that views are not base relations stored in a database but virtual in the sense of being merely “definitions” of queries issued to a database, the updatability of views poses a formidable problem, although there is no problem as long as views are subject to query. While the view update problem has attracted the attention of many scientists and business people, and much research and development has been reported since views were first introduced, a complete resolution to this problem is still an open issue unfortunately. In this paper, a novel concept to resolve this problem is introduced which is called “view updatability based on pro forma guessing of update intention.” First, the updatability of seven basic views, which are defined by using union operation, difference set operation, intersection operation, Cartesian product operation, projection operation, selection operation, and join operation, respectively, is clarified in a systematic manner. Second, based on the result, an algorithm is presented to determine whether a given view, defined arbitrarily by using seven basic view-defining operators recursively, is updatable or not. View updatability, based on pro-forma guessing of update intention, differs essentially from the traditional approach in the sense that certain views actually become updatable in this new approach while they are not updatable in the traditional sense. This is due to the fact that in certain cases the user’s view update intention can be guessed uniquely by checking the “extension” of each view update transformation candidate, which is calculated using temporarily materialized views. Consequently, a unified solution is applied to the view update problem which, until now, had not been thoroughly resolved.

1. はじめに

リレーショナルデータベースビュー(以下，ビュー)は 1974 年にリレーショナルデータモデルを提案したコッド(E. F. Codd)自身により導入された[1]．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るという性質を積極的に用いて導入されたビューは，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し，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り，また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，理論的にも実践的にも多大の関心を集め続けてきた．

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，SQL の用語でいえば導出表，を導く“定義”をビューと称しているにしか過ぎないので，ビューをさらに問合せの対象とするときには問題は生じないが，あくまで定義だけが存在して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が問題となってくる．これを**ビュー更新問題**(view update problem)という．

ビュー更新問題はビューが導入された直後から，多くの研究・開発者の関心を呼んだ興味ある課題であるが，現在まで十分な解は提示されていない．そこで，これまでの研究を通時的に整理してみると次のようになる．まず，この問題は**構文的(syntactic)アプローチ**に基づき研究された．つまり，ビューは実リレーション群がなすデータベース状態

[1] お茶の水女子大学名誉教授
Professor Emeritus of Ochanomizu University
a) yoshi.masunaga@gmail.com

(database state)からビュー状態(view state)への“関数”(function)であると定義することから始まる．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[2]．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．また，図 1 の可換図式に影響を与えるものではないが，$\boldsymbol{u}$ が変換可能な条件として，変換がデータベースに対して余計な更新をかけないこと(no extraneous update)も必要であるが，本報告で示される変換候補は(作り方から)常にその条件を満たしているので，ここでは特段に取り上げない．

したがって，構文的アプローチでは，データベースに課せられた関数従属性などの一貫性制約を考慮しつつ，いかなる基準(criteria)を設定すると正しい変換が可能になるのか，その条件を数学的に明らかにしようとして，これまで多くの研究が報告された[2,3,4,6,7,9,10,11].

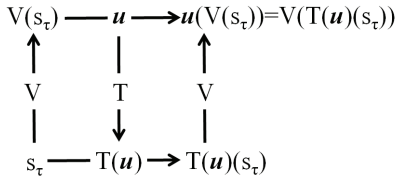

図 1　時刻τにおいてビュー更新要求 $\boldsymbol{u}$ が変換可能であることを示す可換図式

Figure 1　Commutative diagram of representing translatability of view update $\boldsymbol{u}$ at time τ.

しかしながら，ビュー更新問題の研究が進むにつれて，構文的アプローチはその問題解決への基本的枠組みを与えるものではあるが，限界があり，**意味的(semantic)アプローチ**が提唱されることとなった[5]．このアプローチは構文的アプローチで課せられた変換 T の一意性という制約を“緩める”代わりに，代替案の“意味”を明示して更新要求の変換可能性を決定しようとする．つまり，ビュー更新問題は極めて意味的であることが認識・強調されたのである．換言すれば，変換の意味的曖昧性を本質的に除去しようとすれば，ユーザとのやり取りを通して，その更新意図を把握するしか道はないことも明らかになった．実際，ユーザとのインタラクションを組み込み，曖昧性を解消してビューの更新可能性を実装したビュー更新ツール TAILOR が報告されている[8].

上述のように，ビューの更新問題は形式的なアプローチだけにとどまらず，意味的にも難解な問題を孕んでいることが明らかになり，問題解決は積年の課題となって今日に至った．実際，リレーショナルデータベースの現場でも，SQL-89, SQL-92, SQL:1999, SQL:2008 で更新可能なビューが規格化されているが，極めて限定的である[12,13,14,15].

さて，現状では極めて限定的なビュー更新可能性を，これまでにない考え方で打破することができないかと考えた．その解決策が，本報告で提案する“**更新意図の外形的推測に基づくビューの更新可能性**”である．このアプローチは，これまでの構文的・意味的アプローチ，これを**従来型アプローチ**と言おう，ではビューへの更新操作が変換不可能と結論される場合でも，ビューを**一時的にマテリアライズ**してユーザの更新意図を確認し，一方で，その更新要求の変換候補を仮に実行してその結果，これをその変換候補の“**外形**”という，を求めて比較してみると，ユーザが発行した更新要求の意図を一意に“**推測**”できうる場合があり，その時には(ユーザにことさら確かめることはしないで)その変換候補に従って実リレーションの更新を行い，所望のビュー更新を実現させようとするものである．

以下，本報告では，第 2 章でビュー，第 3 章で従来型アプローチによるビューの更新可能性，第 4 章で更新意図の外形的推測に基づいたアプローチによるビューの更新可能性，第 5 章で一般的に定義されたビューの更新可能性判定アルゴリズムを論じる．第 6 章は結論である．

2. ビュー定義とその意味記述

2.1 ビュー定義

ビューを最初に導入したコッド(E. F. Codd)は，ビューをリレーショナル代数表現を用いて定義した[1]．本報告もそれに倣う．ビューに使用するリレーショナル代数演算は次の 7 つとする：和集合演算，差集合演算，共通集合演算，直積演算，射影演算，選択演算，結合演算．ここでは演算の独立性は問わず，よく使われる演算を取り上げた．これら 7 つの基本演算を用いて定義される 7 つの基本ビューは次のとおりである：R と S を和両立(union compatible)な実リレーションとするとき，リレーショナル代数の和集合演算を使って，**和集合ビュー**R∪S が定義される．同様に**差集合ビュー**R－S，**共通集合ビュー**R∩S が定義される．R と S をリレーションとするとき，R×S は**直積ビュー**である．リレーション R(A1, A2, …, An)の属性集合 X = {A1’, A2’, …, Ak’}上の射影演算を使って，**射影ビュー**R[X] が定義される．リレーション R(A1, A2, …, An)の属性 Ai と Aj がθ-比較可能な(θ-comparable)時，R の Ai と Aj 上の **θ-選択ビュー**は R[Ai θ Aj]である．リレーション R(A1, A2, …, An)とリレーション S(B1, B2, . . . , Bm)の Ai と Bj 上の **θ-結合ビュ**

ーは R[Ai θ Bj]S と書ける(ここに Ai と Bj は θ-比較可能).

ビューはこれら 7 つの基本ビュー演算を再帰的に適用して次のように定義される.

[定義 1] (ビュー定義)

1. 実リレーション 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∪V2), (V1−V2), (V1∩V2)もビューである.
3. V1, V2 をビューとするとき, (V1×V2)はビューである.
4. 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.
5. V をビューとするとき, (V[Ai θ Aj])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[Ai θ 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.

定義の中で括弧が使用されているが, それは規則 1 から 6 を再帰的に用いて定義されるビューの構文解析を容易にかつ一意に行う目的で付与している. これは, 再帰的に定義されたビューの更新可能性を判定するアルゴリズムの役に立つ(第 5 章). なお, 自然結合ビューは等結合ビューに準じるので, 便宜上ビュー定義に使用してもよいとする.

2.2 ビューの意味記述

ビュー更新問題に対する意味論的アプローチでは, ビューに対する更新要求 ***u*** の変換候補を求める際に, 基本ビューの意味を拠り所としている[5]. 一般に R をリレーションとするとき, R それ自身を結果リレーションとして返すタップルリレーショナル論理の論理式{t | t∈R}をその**意味**(meaning)と定義することが基本である. したがって, 和集合ビューR∪S の意味は{ t | t∈R ∨ t∈S }である. 表 1 に 7 つの基本ビューの意味をまとめて示す. ビューの意味記述がビューの更新可能性と具体的にどのように関わっていくかは次章で示す.

表 1 7 つの基本ビューとその意味記述

Table 1 Seven basic views and their meaning representations.

ビューの種類	ビューの表記	ビューの意味記述
和集合ビュー	R∪S	{ t | t∈R ∨ t∈S }
差集合ビュー	R−S	{ t | t∈R ∧¬ (t∈S) }
共通集合ビュー	R∩S	{ t | t∈R ∧ t∈S }
直積ビュー	R×S	{ (r, s) | r∈R ∧ s∈S }
射影ビュー	R[X]	{u | u∈dom(A1')×dom(A2')×・・・×dom(Ak')∧(∃t∈R)(t[X] = u)}
選択ビュー	R[Ai θ Aj]	{t | t∈R∧ t[Ai] θ t[Aj]}
結合ビュー	R[Ai θ Bj]S	{(t, u)|t∈R∧u∈S∧t[Ai] θ u[Bj]}

3. ビューの更新可能性—従来型アプローチ—

ビュー更新問題はこれまで構文的・意味的アプローチ, すなわち従来型アプローチのもと精力的に研究されてきた. まず, 確認しておきたいことは, ビューの更新可能性は, ビューは物理的実体のない“定義”だけが存在する仮想的リレーションなので, ビューを更新するということは, もちろん更新要求が発行された時点のデータベースの状態(=インスタンス)によるものの, たとえば,「結合ビューは削除可能か?」といった命題に典型を見るように, 個々の時点で図 1 で示した可換図式が成立するかを問うてきたのではなく, 更新要求が発行されるいかなる時刻においても, 一般的に, 結合ビューが定義されたときに, それに対して発行されたいかなる削除要求も受け付けてよいか?という問題を解こうとしてきたということである. このとき, ビューの更新可能性は次のように定義される.

【定義 2】(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 ***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.

さて, 従来型アプローチのもとで, ビューの更新可能性がどこまで明らかにされてきたかを, 7 つの基本ビューを対象にして示す. その結果は, 表 2 の第 4 列に一覧されるとおりであるが, 若干の補足説明をする.

まず, ビューV に対して更新要求 ***u*** が発行されたとき, ***u*** が変換可能か検証するために変換候補 T を求めなければならないが, そのためにビューの意味記述を使用していることである. 続けて, その具体例を 2 つ示す.

【範例 1】和集合ビューは削除可能

和集合ビューR∪S にタップル t_0 の削除要求 ***d*** = delete t_0 from R∪S が発行されたとしよう. すると t_0 は最早 R∪S の意味を満たさないということだから, R∪S の意味の否定, すなわち¬(t_0∈R)∧¬(t_0∈S) が成立しなくてはならない. これは, ***d*** を R への t_0 の削除要求 ***d1*** = delete t_0 from R と S への t_0 の削除要求 ***d2*** = delete t_0 from S に変換する変換候補 T(***d***)を意味することになる. この変換候補のもと, いかなる時刻においても, またいかなるタップル(群)の削除要求に対しても, このようにして策定される変換候補 T のもとで図 1 の可換図式が成立する. したがって, 和集合ビューは削除可能であると結論付けられる.

【範例 2】自然結合ビューは削除不可能

自然結合ビューへの削除要求を考察してみよう. そこで, 図 2 に示すような, 2 つの実リレーション R(A, B) = {(a, b), (a', b), (a, b')}と S(B, C) = {(b, c), (b, c'), (b', c)}があり, 自然結合ビューR*S = {(a, b, c), (a, b, c'), (a', b, c), (a', b, c'), (a, b', c)}が作られたとする.

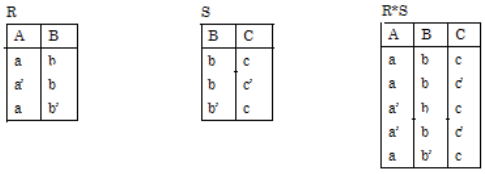

図 2　自然結合ビューR*S のインスタンス
Figure 2　An instance of natural join view R*S.

表 1 に準じ，R*S の意味は次式の通りである：

{(t, v)|t∈R∧v∈dom(C)∧(∃s∈S)(t[B] = s[B]∧v = s[C])}

つまり，“タップル w(= (a, b, c))が R*S の元であるための必要かつ十分条件は(a, b)が R の元でありかつ(b, c)が S の元であること”という意味だと言っている．換言すれば，意味的アプローチによれば，“R*S からタップル w を削除するということは，w が R*S の意味を満たさなくなるということであるから，w はその否定を満たす，つまり(a, b)が R の元でなくなるか，あるいは(b, c)が S の元でなくなるか，あるいはその両方が起こるか，ということと等価である”と言っていることになる．

これにより，たとえば，自然結合ビュー R*S からタップル集合 D1 = {(a, b', c)}を削除したいという要求 ***d*** = delete (a, b', c) from R*S が発行された場合，それを実現するには次に示す 3 つの変換候補があることをすぐに導出できる：

T1(***d***) = delete (a, b') from R
T2(***d***) = delete (b', c) from S
T3(***d***) = do both T1(***d***) and T2(***d***)

さらにもう一つ意味的アプローチが主張していることは，これら 3 つの代替案が実世界で持つ意味的な違いを表現できているということである. つまり, 上記の場合には，確かに T1 でも，T2 でも，あるいは T3 によってでも，副作用を発生させることなく，R*S から D1 を丁度削除することができるが，それらが実世界で持つ意味はまったく異なっているということを明解に示し得ている．つまり，変換候補 T1 が採用されるのは実世界で(a, b')が R という関係性を失った場合であり，変換 T2 が採用されるのは実世界で(b', c)が S という関係性を失った場合であり，そして変換 T3 が採用されるのは実世界で(a, b')が R という関係性と(b', c)が S という関係性を同時に失った場合である．

しかし，この変換の曖昧性を解消するためにはビュー更新要求の背後にある実世界でどのような事象が発生したのかを的確に知る必要があるが,「R*S から D1 を削除したい」というユーザの要求だけでは情報不足でこれを解決するのは無理である．したがって，(この場合)よしんば副作用がないとはいえ変換の一意性が定まらないので，自然結合ビューは削除不可能であると結論付けられる．一般に，結合ビューについても同様な論理が展開できる．

4. 更新意図の外形的推測に基づくビューの更新可能性

表 2 の第 4 列は従来型アプローチに基づくビューの更新可能性の限界を示している．そこでは，直積ビューや結合ビューは削除，挿入，書換はすべて不可能である．また，書換操作は，削除とそれに続く挿入操作の系列と捉えているから，ビューが削除不可能あるいは挿入不可能ならば直ちに書換不可能と結論されてきた．本報告で提案する“更新意図の外形的推測に基づくビュー更新可能性”と“直接書換法”はこのような限界を打破するべく考案された解決策である．

4.1 更新意図の外形的推測に基づくビューの更新可能性の考え方

この新しい考え方は，従来型アプローチを踏襲しつつも，直積ビューや結合ビューが更新不可能となる原因を究明し，ビューの更新可能性を，ヒューマンインタラクションに頼ることなく，極限にまでに高めることを意図している．

3 章の範例 2 で示したように，従来型アプローチにおいて，たとえばなぜ自然結合ビューが削除不可能と結論されるかといえば，変換に 3 つの代替案が生じ，その意味的曖昧性をヒューマンインタラクションなしでは解消できないと結論したためであった．そこで，このような曖昧性が生じる状況をさらに詳細に分析してみることとした．その結果，意味的曖昧性が生じても，更新要求が発行された時刻とその時のデータベースの状態によっては，ビュー(と中間ビュー)を一時的にマテリアライズし，更新要求と変換候補を仮に実行してみると，その“外形”(extension)からユーザが発行した更新要求の意図を一意に“推測”できうる場合があり，その時はその変換でビュー更新を行ってよい，とする考え方に行き着いた．この典型例を次に示す．

【範例 3】自然結合ビューは外形的推測に基づき削除可能

図 2 に示した自然結合ビューR*S に対して，タップルの集合 D 2= {(a, b, c), (a, b, c')}の削除要求 ***d*** = delete D2 from R*S が発行されてきた場合を考えてみる．この要求に対して，意味的アプローチに基づき変換候補を求めると，次に示す 3 つの代替案があることがわかる：

T1(***d***) = delete (a, b) from R
T2(***d***) = delete {(b, c), (b, c')} from S
T3(***d***) = do both T1(***d***) and T2(***d***)

この曖昧性から従来型アプローチでは，表 2 第 4 列に示したとおり，削除不可能と結論されている．しかし，ここで，R*S をマテリアライズし ***d*** を実行すると共に，T1 を仮に実行し，その結果リレーションと S を自然結合して，再びビューをマテリアライズして変換候補 T1 の外形を求めてみると，それは R*S－D2 となり(－は差集合演算)，マテリアライズされた R*S に対する ***d*** の実行結果と同じとなることを知る．一方，T2 や T3 を仮に実行すると外形は共

に R*S－{D2∪{(a', b, c), (a', b, c')}}となり，副作用が発生することが分かる．つまり，唯一 T1 を実行した場合に限り図 1 の可換図式が成立することがこの試算で分かり，我々はこの状況を「ユーザがなぜ自然結合ビューR*S に対して D2 の削除要求を発行してきたのか，その意図は(本人に聞く以外には)知る由もないが，ユーザが要求しているタップル群の削除を丁度実現できる変換候補がただひとつだけある(これが T1)ので，きっと実世界で T1 に該当する事象，つまり R からタップル(a, b)を削除することに相当する事象が発生して D2 の削除要求になったのであろうと**推測**(guessing)して，このようにビュー更新を行う」こととした．このとき，R*S への削除要求 ***d*** は外形的推測に基づき変換可能である，一般的に自然結合ビューは外形的推測に基づき削除可能であるという．

なお，外形的推測を行ったからといって，必ず変換の曖昧性が解消されるわけではないことは当然で，先に示した範例 2 はそのような例である．

また，和集合ビューに対するタップル(群)の挿入要求，差集合ビューからのタップル(群)の削除要求，共通集合ビューからの削除要求は変換の曖昧性が生じるが，これらは外形を計算しても解消できないので，本質的に更新不可能である．したがって，差集合ビューや共通集合ビューへのタップル(群)の書換要求も本質的に不可能である．

4.2 更新意図の外形的推測に基づく 7 つの基本ビューの更新可能性

先述のとおり，更新意図の外形的推測に基づくビューの更新可能性は，従来型アプローチの結果を基にしているので，従来更新可能とされたビューはこの新しいアプローチのもとでも更新可能である．逆に，従来型アプローチで更新不可能となっているビューと更新要求の組合せのうち，どこまでがこの新しいアプローチで更新可能となるかが問題である．そこで，そのような組合せ 12 個の中から範例を幾つか示し，全体の結果をまとめて表 2 の第 3 列に示す．

まず，書換を削除と挿入の系列と捉えることによる限界を打破するために，“直接書換法”を導入する．

【範例 4】和集合ビューは直接書換可能

書換の対象となったタップル(群)を削除してしまうことなく，直に旧値を新値に書き換えて操作を終了する更新法を**直接書換法**と言おう．これは，商用の DBMS ではもう少し工夫されているが，実質的に執り行われている手法である．書換を削除とそれに続く挿入という 2 つの独立した更新操作の系列と見做す従来型アプローチでは，和集合ビューの書換は，和集合ビューは削除可能であるが挿入不可能なので，書換不可能と判定される．しかし，直接書換法に従えば，和集合ビューは直接書換可能である．射影ビューにも適用でき，射影集合ビューは直接書換可能である．

次に，従来型アプローチでは，直積ビューや結合ビューは一切更新不可能であった点について考察する．

【範例 5】直積ビューは外形的推測に基づき挿入可能

一般性を失うことなく，R = {(a1, b1), (a2, b2)}, S = {(c1, d1), (c2, d2)}とする．図 3 に R と S，それらの直積ビューV = R×S のインスタンスを示す．そこで，V に，タップルの集合 I1 = {(a, b, c, d)}の挿入要求 ***i1*** = insert I1 into R×S が発行されたとする．V の意味から，(a, b)∈R ∧ (c, d)∈S が成立しないといけないので，この挿入要求を実現する唯一の変換候補 T(***i1***) = {insert (a, b) into R, insert (c, d) into S}が得られる．しかし，それに伴い V には{(a, b, c1, d1), (a, b, c2, d2), (a1, b1, c, d) , (a1, b1, c, d) }も挿入されてしまい，従来型アプローチが示しているように挿入不可能である．

しかし，もし V に I2 = {(a3, b3, c1, d1), (a3, b3, c2, d2)}を挿入したいという要求 ***i2*** があったとすると，V の意味から，変換候補 T(***i2***) = {insert (a3, b3) into R，insert {(c1, d1), (c2, d2)} into S} が得られる．このとき，T の外形を計算してみると，(R×S)∪I2 となり，この挿入要求は受理されてよい．ここで，S へのタップル集合の挿入は，それらが S に既存なので，S に何らの変化を引き起こさないことに注意する．つまり，V に I2 を挿入したいという要求は実世界で R に(a3, b3)を挿入しないといけない事象が発生したから発行されたのだろうと推測できる．つまり，直積ビューは外形的推測に基づき挿入可能である．なお，この場合，外形計算は範例 3 で示したような変換候補の曖昧性を除去するために機能したのではなく，変換候補はただ一つしかないのだが，副作用が生じないことを確認するために機能している．

R

A	B
a1	b1
a2	b2

S

C	D
c1	d1
c2	d2

R×S

A	B	C	D
a1	b1	c1	d1
a1	b1	c2	d2
a2	b2	c1	d1
a2	b2	c2	d2

図 3　直積ビューR×S のインスタンス

Figure 3　An instance of Cartesian product view R×S.

【範例 6】直積ビューは，書換要求が両立していれば，外形的推測に基づき書換可能

範例 5 と同じく，図 3 に示した直積ビューV = R×S を考える．まず，V のタップル群 U = {(a1, b1, c1, d1), (a1, b1, c2, d2)}の 書換要求 ***r1***= rewrite {(a1, b1, c1, d1), (a1, b1, c2, d2)} of R×S to {(a3, b3, c1, d1), (a3, b3, c2, d2)} が発せられたとする．範例 3 で扱った自然結合ビューの場合と同様に，直積ビューV は U に関して外形的推測に基づき削除可能であり，この時の副作用のない一意な変換は T1(***d1***) = delete (a1, b1) from R である．続いて，書換要求を実現させるためには，V の意味から，(範例 5 に同じく)対応する挿入要求 ***i2*** が実現されねばならないが，これは変換 T(***i2***)により実現できることが分かる．このとき，変換 T1(***d1***)と T(***i2***)は書換要求 ***r1*** に関して**両立**(compatible)している，あるいは単に書換要求 ***r1*** は両立している，ということにする．

しかし, U の書換要求 ***r2*** = rewrite {(a1, b1, c1, d1), (a1, b1, c2, d2)} of R×S to {(a1, b1, c3, d3), (a1, b1, c4, d4)} が発せられた場合には，上記の変換 T1 のもと V は U に関して外形的推測に基づき削除可能であるものの，V の意味をもとに，***i3*** = insert {(a1, b1, c3, d3), (a1, b1, c4, d4)} into R×S を実現しようとすると，T(***i3***) = {insert (a1, b1) into R, insert {(c3, d3), (c4, d4)} into S}が実行されねばならないことになる．しかし，これが実行されると外形は V∪{(a1, b1, c3, d3), (a1, b1, c4, d4), (a2, b2, c3, d3), (a2, b2, c4, d4)}となり，副作用が発生して不都合となる．これは，T(***d1***)と T(***i3***)が ***r2*** に関して両立していなかったから発生した現象である．

つまり，直積ビューに書換要求が発行された場合，まず書換の対象となったタップル群に対して外形的推測に基づき削除可能か検証する．もしそうならば，続いて行われる挿入操作が先行して行われた削除操作と両立しているか検証し決まる．つまり，直積ビューは，書換要求が両立していれば，外形的推測に基づき書換可能である．結合ビューの書換要求についても，全く同様な議論が展開される．

以上，更新意図の外形的推測に基づくビューの更新可能性の要点を述べた．表 2 の第 3 列に結果をまとめる．その結果，従来型アプローチと本報告で提案した新しいアプローチのビュー更新可能性の違いが一目瞭然となっている．

表 2　7 つの基本ビューの更新可能性の比較一覧

Table 2　List of updatability of 7 basic view definitions.

基本ビューの種類	更新操作	更新意図の外形的推測に基づく更新可能性	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	□	×
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	□	○(主キーを含む)
選択ビュー	削除	○	○
	挿入	○	○
	書換	○	○
結合ビュー	削除	◎	×
	挿入	◎	×
	書換	◎(書換要求両立)	×

○，×：従来型アプローチで更新可能，不可能．◎：更新意図の外形的推測に基づくアプローチで更新可能．□：直接書換法で書換可能．×*：本質的に不可能，を表す．

5. 一般的に定義されたビューの更新可能性

ビューは定義 1 に示された通り 7 つの基本演算を再帰的に適用されて定義される．本章では，このように一般的に定義されたビュー，単にビューという，の更新可能性をどのようにして判定していくか，その方法を論じる．

5.1 ビュー定義木とデータ構造

ビューの定義を与えた定義 1 では，括弧付でビューを定義する体系としたので，一般的に定義されたビューが与えられるとその構文解析は紛れることなく行われ，ビュー定義木(view definition tree)が得られる．7 つのビュー演算は単項あるいは 2 項の演算子であるので，ビュー定義木はビューを根とする 2 分木(binary tree)となる．2 項演算子の差集合演算，直積演算，結合演算子の存在から，この木は順序木(ordered tree)となる．したがって幅優先探索(breadth-first search)が可能である．

そこで，ビューの更新可能性判定アルゴリズムをプログラミング可能とするために，ビュー定義木のノード(node)のデータ構造を図 4 の通りとする．VNAME フィールドにはノードの識別子，VDEF フィールドにはそのノードを定義するために使われた演算を具体的に格納する(ノードが葉の場合 Base とする)．LLINK フィールド，RLINK フィールドにはそのビューを定義するために使われた中間ビューや実リレーションへのポインタを格納する(それらがない場合 NULL とする)．

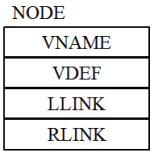

図 4　ビュー定義木のノードのデータ構造

Figure 4　Data structure of a node of view definition tree.

ここで，具体的に範例 7 で(至極一般的と考えられる)ビュー **SE@横浜** を定義し，そのビュー定義木を表す 2 分木のデータ構造を図 5 に示す．

【範例 7】ビュー SE@横浜とビュー定義木

実リレーションとして，社員(社員番号，社員名，所属，職種)と部門(部門番号，部門名，所在地)があり，それ上でビュー SE@横浜 が次のように定義されたとする：

SE@横浜 = (((社員[職種 = ‘SE’])[所属 = 部門番号]
(部門[所在地 = ‘横浜’]))[社員番号，社員名，所属])

ここで，中間ビュー1=((社員[職種 = ‘SE’])[所属 = 部門番号](部門[所在地 = ‘横浜’])), 中間ビュー2=(社員[職種 = ‘SE’]), 中間ビュー3= (部門[所在地 = ‘横浜’])である．なお，実リレーションのノードと実リレーションは同一視する．

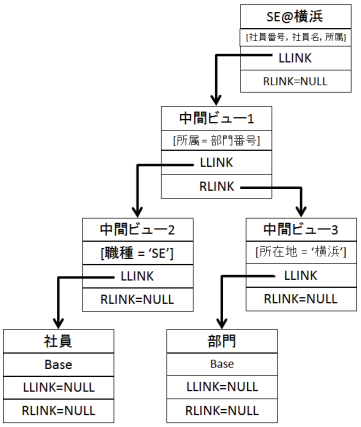

図 5　ビュー SE@横浜 を表す 2 分木のデータ構造
Figure 5　Data structure of a binary tree representing view SE@Yokohama.

5.2 更新意図の外形的推測に基づくビューの更新可能性判定アルゴリズム

【アルゴリズム D】更新意図の外形的推測に基づくビューの更新可能性判定アルゴリズム

V をビューとし，***u*** を V に発行された更新要求とする．
UREQ(P) = NODE(P)への更新要求,
UREL(P) = NODE(P)の更新要求への関連度，とする．

D1. [初期化] V のビュー定義木を作る．P ← 0 (零), UREQ(0) = ***u***, UREL(P) = 1 (1 は“関連有”，0 は“関連無”)，ここに 0 ≦P≦N_{max} と設定する(ポインタ変数 P はビュー定義木の根を指す時 0，順次幅優先探索順に 1, 2, …, N_{max} (N_{max}≧0)と移動する).

D2. [終了検出] もし P = N_{max} なら，D8 に行く．そうでなければ，もし UREL(P) = 1 なら, D3 に行く． もし UREL(P) = 0 なら, P ← P+1 として D2 に戻る．

D3. [表参照] 表 2 の第 3 列を参照し，対 VDEF(P) – UREQ(P) の更新可能性を見る．もし“○”，“◎”，“□”なら，それぞれ D4，D5，D6 に行く．それ以外は，D7 に行く．

D4. [従来型でのビュー更新変換] もし VDEF(P) = ∪，－，∩なら，UREQ(P)を LLINK(P)と RLINK(P)でリンクされた部分木の根に対する更新要求 UREQ(LLINK(P)) と UREQ(RLINK(P))に然るべく変換し，P ← P+1 として D2 に戻る．もし，VDEF(P) =選択演算なら，UREQ(LLINK(P))を然るべく設定し，P ← P+1 として D2 に戻る. もし VDEF(P) = 射影演算で UREQ(P) = 削除要求なら, UREQ(LLINK(P))を然るべく設定し, P ← P+1 として D2 に戻る．もし UREQ(P) = 挿入演算なら，射影が主キーを含めば UREQ(LLINK(P))を然るべく設定し, P ← P+1 として D2 に戻る. 含まなければ, D7 に行く．

D5. [更新意図の外形的推測に基づくビュー更新変換] UREQ(P) = 削除要求か挿入要求の場合，NODE(P)を一時的にマテリアライズし，UREQ(P)を実行する．その結果に基づき UREQ(P)を補完し，その UREQ(P)を UREQ(LLINK(P))と UREQ(LLINK(P))に変換する候補 T1，T2，T3 を求める．次に，NODE(LLINK(P)) と NODE(LLINK(P))を一時的にマテリアライズし，それら使って各候補について外形を計算する．もし，T1 でのみ更新可能なら，RLINK(P)でリンクされた部分木の全てのノードに対して UREL(P) = 0 と設定する．もし T2 のみなら，LLINK(P)でリンクされた部分木の全てのノードに対して UREL(P) = 0 と設定する．もし T3 のみなら UREL(P)の操作はしない． P ← P+1 として D2 に戻る. もし更新可能でなければ, D7 に行く．UREQ(P) = 書換要求の場合，同様に処理をするが，もしその両立性が成立しなければ，D7 に行く．

D6. [直接書換でのビュー書換変換]

VDEF(P) = ∪の場合，直接書換法により UREQ(P)は書換要求 UREQ(LLINK(P))と UREQ(RLINK(P))に変換される．VDEF(P) = 射影演算の場合，直接書換法により UREQ(P)は書換要求 UREQ(LLINK(P))に変換される．共に，P ← P+1 として D2 に戻る．

D7. [更新不可能] V は ***u*** に関して，更新可能でない．

D8. [更新可能] V は ***u*** に関して，更新可能である．実リレーション群への更新は，関係した更新変換を合成して得られる．

以下，このアルゴリズムの動作を，ビュー SE@横浜 にある削除要求が発行された場合を想定してシミュレーションした結果の概略を示す．

【範例 8】ビュー SE@横浜の更新可能性判定

ビュー SE@横浜に使われた実リレーション社員と部門のインスタンスは図 6 に示された通りとし，そのとき，ビューに対して発行された削除要求 ***d*** は次の通りとする：

d = delete (003，鈴木，K41) from SE@横浜

(1) (ステップ D1)　SE@横浜のビュー定義木を作成する(図 5). 幅優先探索順にノードを索引付する．すべての P(この例では，0～5 = N_{max})に対して UREL(P) = 1 と設定する．P = 0, UREQ(0) = ***d*** と置いて，D2 に行く．

(2) (ステップ D2)　P = 0 ≠ N_{max} かつ UREL(0) = 1 なので，D3 に行く．

(3) (ステップ D3)　VDEF(0)は，NODE(0) = SE@横浜 が中間ビュー1 の[社員番号，社員名，所属]上の射影ビュー

であることを示している．対 VDEF(0)－UREQ(0)を表 2 第 3 列で表参照すると，項目は“○”なので，D4 に行く．

(4) (ステップ D4)　VDEF(0)は射影演算で，UREQ(0) = 削除要求なので，UREQ(0)を UREQ(LLINK(0)) = UREQ(1) = ***d1*** = delete {(003，鈴木，K41, *, *, *, *)} from 中間ビュー1　に変換する．ここに，* は任意の値か NULL を表す．P ← P+1 として D2 に戻る．

(5) (ステップ D2)　P = 1 ≠ N_{max} かつ UREL(1) = 1 なので，D3 に行く．

(6) (ステップ D3)　VDEF(1)は，NODE(1) = 中間ビュー1 が中間ビュー2(= NODE(2))と中間ビュー3(= NODE(3))の[所属 = 部門番号]上の結合ビューであることを示している．対 VDEF(1)－UREQ(1)を表 2 第 3 列で表参照すると，項目は“◎”なので，D5 に行く．

(7) (ステップ D5)　この削除可能性は更新意図の外形的推測によってなされる．UREQ(1) = 削除要求 なので，NODE(1)をマテリアライズし(図 7)，UREQ(1)を仮に実行する．その結果，中間ビュー1 からタップル(003，鈴木，K41, SE, K41, AI, 横浜)が削除されることを知る．それにより，UREQ(1)を次のように補完できる：UREQ(1) = ***d2*** = delete {(003，鈴木，K41, SE, K41, AI, 横浜)} from 中間ビュー1．続いて，中間ビュー2 と中間ビュー3 を一時的にマテリアライズし，***d2*** の 3 つの変換候補を実行して，それらの外形が受け入れられるか否か判定する．ここに，3 つの変換候補は次の通りである：

T1(***d2***) = delete {(003，鈴木, K41, SE)} from 中間ビュー2

T2(***d2***) = delete {(K41, AI，横浜)} from 中間ビュー3

T3(***d2***) = do both T1(***d2***) and T2(***d2***)

その結果，T1(***d2***)のみが図 1 の可換図式を成立させることが分かる．そこで，RLINK(1)でリンクされている部分木の各ノード P に対して，UREL(P) = 0 とする．P ← P+1 (従って P = 2)として，ステップ D2 に戻る．

続けて，ステップ D2, D3, D4, D2, D2, D3, D2, D8 と実行することにより，SE@横浜は ***d*** に関して削除可能である．

社員

社員番号	社員名	所属	職種
001	佐藤	K41	NE
003	鈴木	K41	SE
006	高橋	K41	SE
007	ボンド	K55	SE
010	田中	K55	NE
013	ゴルゴ	K81	SE

部門

部門番号	部門名	所在地
K41	AI	横浜
K55	DB	横浜
K81	PL	東京

図 6　実リレーション社員と部門のインスタンス

Figure 6　Instances of base relations EMP and DEPT.

中間ビュー1

社員番号	社員名	所属	職種	部門番号	部門名	所在地
003	鈴木	K41	SE	K41	AI	横浜
006	高橋	K41	SE	K41	AI	横浜
007	ボンド	K55	SE	K55	DB	横浜

図 7　一時的にマテリアライズされた中間ビュー1

Figure 7　Temporarily materialized intermediate view 1.

6. おわりに

本報告では，これまでその全貌が明らかになっていなかったリレーショナルデータベースのビュー更新問題を“更新意図の外形的推測に基づく更新可能性”という考え方を新たに導入することにより解決した．今後，処理コストの最適化を考慮しつつ，現在は極めて限定的な国際標準リレーショナルデータベース言語 SQL のビューサポート機能の実質的な拡張に向けて研究･開発を展開する予定である．

謝辞　有益なご意見をいただいた石井達夫氏(SRA OSS, Inc.日本支社)に感謝する．

参考文献

1) Codd, E., Recent investigations in a relational database system, Information Processing 74, pp. 1017-1021, North-Holland, 1974.
2) Dayal, U. and Bernstein, P., On the updatability of relational views, Proc. 4th VLDB, pp.368-377, 1978.
3) Bancilhon, F. and Spyratos, N., Update semantics of relational views, ACM TODS, Vol.6, No.4, pp.557-575, 1981.
4) Cosmadakis, S. and Papadimitriou, C., Updates of relational views, ACM PODS, pp.317-331, 1983.
5) Masunaga, Y., A relational database view update translation mechanism, Proc. 10th VLDB, pp.309-320, 1984.
6) Keller, A., Algorithms for translating view updates to database updates for views involving selections, projections, and joins, ACM PODS, pp.154-163, 1985.
7) Keller, A., Choosing a View Update Translator by Dialog at View Definition Time, Proc. 12th　VLDB, pp.467-474, 1986.
8) Sheth, A., Larson, J., and Watkins, E., TAILOR, A Tool for Updating Views, LNCS, Vol. 303, pp.190-213, Springer, 1988.
9) Gottlob, G., Paolini, P., and Zicari, R., Properties and update semantics of consistent views, ACM TODS, Vol.13, No.4, pp.486-524, 1988.
10)　Langerak, R., View updates in relational databases with an independent scheme, ACM TODS, Vol.15, No.1, pp.40-66, 1990.
11)　Chen, I.-M, Hull, R., and McLeod, D., An execution model for limited ambiguity rules and its application to derived data update, ACM TODS, Vol.20, No.4, pp.365-413, 1995.
12)　Melton, J. and Simon, A., SQL:1999 Understanding Relational Language Components, 893p., Morgan Kaufmann, 2002.
13)　増永良文，リレーショナルデータベース入門[新訂版]，361p., サイエンス社，2003
14)　土田正士，小寺孝，SQL2003 ハンドブック，455p., ソフト・リサーチ・センター，2004.
15)　PostgreSQL 9.3.9 Documentation, Chapter 34. The Information Schema, 34.63. Views, http://www.postgresql.org/docs/9.3/static/infoschema-views.html.