

連合データベースシステムにおける導出データの一貫性管理法

2F-3

木實新一 古川哲也

九州大学大型計算機センター

1. まえがき

データベースシステム(DBS)の形態として、集中型、分散型、連合型、非連合型がある。前の二つの型のDBSは、単一のデータベース管理システム(DBMS)がデータベースを管理するが、後の二つの型のDBSは複数のDBMSから構成されており、マルチDBSと呼ばれる。

データベースが世に現れてから現在までの間に、機械可読なデータ資源が様々な形で蓄えられてきた。今後も、二次記憶の低価格化、安価で高性能なワークステーションの普及などにより、世界中で運用されるDBSの絶対数や蓄積されるデータ資源の量は増加を重ねていくと思われる。このような状況ですます重要となるのは、複数のDBSに対するデータ操作機能を持つマルチDBSを提供し、冗長な大量データの効率よい利用を促進することである。

マルチDBSに含まれるそれぞれのDBSを要素DBSと呼ぶ。マルチDBSには連合型のもので非連合型のものがあるが、非連合型のもは要素DBSが均一であることを要求し、自律的な要素DBSを取り扱うことができない。本稿では、連合型のマルチDBS(連合データベースシステム)を議論の対象とする。すなわち、要素DBSの分散性、異種性だけでなく、自律性も考慮した、マルチDBSにおける導出データの一貫性管理法について論じる。

まず、要素DBSの分散性、異種性、自律性を考慮した更新処理のモデルについて論じる。次に、具現化した導出データを局所的な一貫性管理が可能となるように配置する方法を示し、さらに、連合データベースシステムを、更新処理に関して互いに自律的なグループに分割して一貫性管理を効率化する方法について述べる。

2. 連合データベースシステムの特徴

連合データベースシステム(連合DBS)は、複数のDBSと、それらの間の調整を行うソフトウェアである連合データベース管理システムからなる。集中型のDBSは一般に3層のスキーマ構造を持つが、連合DBSは要素DBSを複数包含するため、従来のような3層のスキーマ構造では記述できない。本稿では、文献[1]に基づいた5層のスキーマ構造を採用する(図1)。

局所スキーマ(LO)は要素DBSの概念スキーマである。要素スキーマ(EL)は局所スキーマを共通データモデルに変換したものであり、ここで要素データベースの異種性を吸収する。輸出スキーマ(XP)は要素スキーマから連合に参加する部分を取り出したものであり、自律性の管理を支援する。連合スキーマ(FE)は複数の輸出スキーマを統合したものであり、分散性の管理を支援する。外部スキーマ(XT)は連合スキーマから、利用者やアプリケーションに見せる部分を取り出したものである。

連合DBSにおいて重要な3つの概念として、分散性、異種性、自律性がある。

分散性 連合DBSが管理する複数のDBSは、複数の計算機システムに分散して管理される。この論文では、分散性を考慮して、導出データの管理の一貫性を保つための方法を提案する。論文のタイトルは「Keeping Consistencies of Materialized Derived Data in Federated Database Systems」
Shin'ichi KONOMI, Tetsuya FURUKAWA
Kyushu University

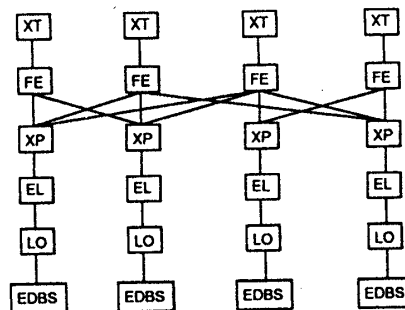


図1: 連合データベースシステムのスキーマ構造

システムに分散していてもよい。また、連合DBSが管理するDBSは分散DBSや連合DBSであってかまわない。

異種性 要素DBSのデータベース管理システムは異なっていてよい。また、ある要素DBSのスキーマ要素と、別のDBSの構文的に等価なスキーマ要素が意味的に異種であって構わない。

自律性 各要素DBSを自由に設計することができる。また、要素DBS内部で発行された演算の実行は、外部から発行された演算の影響を受けない。さらに、他の要素データベースの資源は自由に共有できる。

3. 分散性、異種性、自律性を考慮した一貫性管理のモデル

まず、更新従属性を定義する。更新従属性は連合DBSの自律性と関連が深い。

[定義1] 同一実体を表す全てのデータの集合を E とする。 ■

[定義2] データ $x_1 \in E$ に続いて、データ $x_2 \in E$ を x_1 と同じ値に変更しなければならないとき、これを $x_1 \rightarrow x_2$ と表す。 ■

[定義3] $N = \{x \mid \forall x \in E\}$, $B = \{(x_i, x_j) \mid \forall x_i, x_j \in E, \text{if } x_i \rightarrow x_j\}$ とする。グラフ $G(N, B)$ の強連結成分の集合を $\{E_i\}$ で表し、 E_i を均一値集合と呼ぶ。 ■

[定義4] グラフ $G(N, B)$ において、均一値集合 E_i の要素から E_j の要素へ向かうパスがある場合、均一値集合 E_j は E_i に従属するといひ、これを $E_i \rightarrow E_j$ で表す。 ■

$E_i \rightarrow E_j$ を更新従属性と呼ぶ。

データ x_1, x_2, \dots, x_m から、集約関数 f によって導かれるデータを x とする。すなわち、

$$x = f(x_1, x_2, \dots, x_m), x_i \in E (1 \leq i \leq m)$$

集約関数により連合DBSの異種性の一部を解決できる。

f により導出されるデータ x を実際にデータベースのデータとして保持することを具現化と呼ぶ。具現化は x の効率良い検索を可能にするが、 $x_i (1 \leq i \leq m)$ が更新されるとそれに応じて具現化した x を更新しなければならない。

文献[2]に示された、時間的な遅れを許容する一貫性制約を基に、連合DBSの一貫性制約を分散性を考慮して次のように定義する。

[定義 5] $\forall x_1, x_2 \in E_i, t \geq 0, 0 \leq k \leq \alpha$ について, $x_1(t) = x_2(t-k)$ または $x_1(t-k) = x_2(t)$ となる k が存在するとき, 均一値集合 E_i は最大遅延 α で一貫しているという. ■

上記の一貫性制約を $W(E_i) = \alpha$ で表す.

[定義 6] 集約関数 $f, x = f(x_1, x_2, \dots, x_m), t \geq 0, 0 \leq k_i \leq \beta, (1 \leq i \leq m)$ について, $x(t) = f(x_1(t-k_1), x_2(t-k_2), \dots, x_m(t-k_m))$ となる $k_i, (1 \leq i \leq m)$ が存在するとき, 具現化された x は最大遅延 β で一貫しているという. ■

上記の一貫性制約を $W(x) = \beta$ で表す. 最後に, 一貫性管理コストの日安として次の値を定義しておく.

[定義 7] 一貫性制約の集合を C, x を具現化した導出データとする. x を C に関して一貫したものとするために読み書きする必要があるデータを持つ要素 DBS の数を $PE(C, x)$ とする. ■

4. 具現化した導出データの効率良い一貫性管理法

本節では, まず $PE(C, x) \leq 2$ となるように $x = f(x_1, x_2, \dots, x_m)$ を具現化する方法について述べる. ただし, f は集約関数とする.

[定義 8] f を関数, $y = f(x_1, x_2, \dots, x_i, \dots, x_s), y' = f(x_1, x_2, \dots, x'_i, \dots, x_s), x_i \neq x'_i$ とする. 任意の $i (1 \leq i \leq s)$ について, $y' = g(y, x_i)$ または $y' = g(y, x'_i)$ となる関数 g が存在するとき, f を局所更新関数と呼ぶ. ■

[定理 1] f が局所更新関数である場合, 具現化した導出データ $x = f(x_1, x_2, \dots, x_m)$ は, x_i を挿入, 削除, 変更した場合, 一貫性制約 $C = \{W(x) = \beta\}$ に対して, $PE(C, x) \leq 2$ である. ■

$PE \leq 2$ であっても, 2 計算機システム間の通信量が大きくては処理コストが大きくなってしまいが, 局所更新関数で導出したデータの更新に必要なのは, 変更分のデータのみであるため, 通信コストは極小である.

$f(x_1, x_2, \dots, x_m)$ が局所更新関数でない場合, f を関数 h_1, h_2, \dots, h_t を引数にとる関数 $f'(h_1, h_2, \dots, h_t)$ に変形し, $h_i (1 \leq i \leq t)$ が局所更新関数であれば, x を具現化している計算機システムに h_i の値も具現化する. これを繰り返し行なうことで, f が局所更新関数でなくとも定理 1 が成り立つ.

[例 1] モード (最頻値) は算術平均, 中央値と並ぶ代表的な統計量であり, ここではモードを集約関数 $mode$ を例にとる. $mode$ は局所更新関数ではない. 図 4. では, 6 つの要素 DBS, $EDBS_1, EDBS_2, EDBS_3, EDBS_4, EDBS_5, EDBS_6$ にデータ $x_1, x_2, x_3, x_4, x_5, x_6 \in E$ を保持している. また, $EDBS_5$ には $x = mode(x_1, x_2, x_3, x_4) = 5$ を具現化したものを保持している. $x_4 = 5$ を $EDBS_4$ から削除した場合の x の一貫性管理は, $EDBS_1, EDBS_2, EDBS_3$ それぞれに対し, x_1, x_2, x_3 の値を求めるための演算を発行し, 求められた値からモードを計算し直すという方法で行なう.

ところで, ある値の度数を求める関数 $freq$ 局所更新関数であり, $mode(x_1, x_2, \dots, x_m) = \max_{1 \leq i \leq m} freq(x_i)$ である. $freq$ の値 $freq(5) = 3, freq(2) = 1$ を x と同じ EDB_5 に具現化していれば, $x_4 = 5$ を $EDBS_4$ から削除した場合の x の一貫性管理は, 5 が 1 つ削除されたことを EDB_5 に伝え, $freq(5) = 2$ とするだけでよい. ■

次に, 均一値集合 E_i の一貫性管理と, 更新従属性の管理のための処理を考慮して, 効率良く x の一貫性管理を行なう方法を示す.

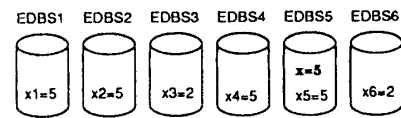


図 2: 具現化したモード値の一貫性管理

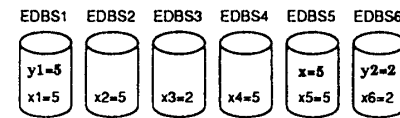


図 3: 具現化したモード値 \hat{x} の一貫性管理

[定義 9] 均一値集合 E_i において, 最も最近に更新した値 $x \in E_i$ を E_i の代表値とする. ■

[定義 10] $x = f(x_1, x_2, \dots, x_m)$ を集約関数 f から導出される具現化されたデータ. $x_i \in E_j (1 \leq i \leq m)$ となる E_j の代表値の集合を R とする. $\hat{x} = f(y_1, y_2, \dots, y_s), y_i \in R$ は代表値から導出されるデータである. ■

x_i が更新された場合, E_j の要素を全て等しくするために $\#(E_j)$ 個のデータが更新され, 導出データ x が $W(x) = \beta$ を満たすための処理を最悪 $\#(E_j)$ 回行なわねばならなくなる. これを避けるためには, x の代わりに \hat{x} を具現化するとよい. \hat{x} が $W(\hat{x}) = \beta$ となるための処理の手間は, x が $W(x) = \beta$ となるための処理の手間の約 $1/\#(E_j)$ 倍である.

[例 2] 図 4. において, $E_1 = \{x_1, x_2, x_4, x_5\}, E_2 = \{x_3, x_6\}$ を均一値集合とする. $x_4 = 5$ を 4 に変更した場合, x_1, x_2, x_5 もそれぞれ 4 に変更していく. この過程で, $x = mode(5, 5, 2, 4), x = mode(4, 5, 2, 4), x = mode(4, 4, 2, 4)$ を毎回計算して x を更新しなければならない. 図 4. では, E_1, E_2 の代表値 y_1, y_2 を $EDBS_1$ と $EDBS_6$ にそれぞれ保持している. $x_4 = 5$ を 4 に変更した場合, y_1 を 5 から 4 に変更し, x_1, x_2, x_5, x を y_1 に基づいてそれぞれ 5 から 4 に変更する. ■

5. むすび

連合 DBS の分散性, 異種性, 自律性を考慮した一貫性管理のモデルの上で, 具現化した導出データの一貫性管理を効率良く行なう方法を示した.

均一値集合の代表値を用いる方法では, 均一値集合内の更新波及に伴う導出データの一貫性管理の手間を省くことができるだけでなく, 更新が生じてから導出データの一貫性が保たれるまでの遅延時間を小さくすることができる. また, 更新従属性の推移的閉包に対して代表値を持たせる方法も考えられる.

局所更新関数の性質を更に明らかにし, 関数を局所更新関数に分解するアルゴリズムを示すことが必要であろう.

参考文献

[1] A. P. Sheth and J. A. Larson, *Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases*, ACM Computing Surveys, Vol.22, No.3, pp.183-236, Sept. 1990.
 [2] R. Alonso, D. Barbara, and H. Garcia-Molina, *Data Caching Issues in an Information Retrieval System*, ACM Trans. on Database Systems, Vol. 15, No.3, pp.359-384, Sept. 1990.