D-15

分散型データベースにおけるアクセス性能の効率化
Efficiency of Access on Distributed Databese

進谷　洋平
Yohei Shintani

1. まえがき

関係データベースは多くの業務で利用されることが多く、全てのデータベース更新に対して不整合が生じないようにするためにデータベーススキーマを正規化する必要があった。しかし、データベーススキーマを第3正規形にすると、関連するデータが複数の関係スキーマに分けられる事が多く、検索処理の際に基底関係を結合する必要がある。

また分散型データベースでは複数にデータベースサーバが分散しているため、クライアントがあるデータを問い合わせ、サーバにそのデータがない場合、他のデータベースサーバに問い合わせなければならない。よって、検索処理の際に、集中型データベースに比べ、検索処理の処理時間が長くなる可能性があるという欠点がある。

よって、この結合処理は、分散データベースで特にレコード数の多いデータベースにおいて、処理時間が長くなるという問題がある。

これに対して、結合処理の対象になることの多い基底関係をあらかじめ結合して、各々のデータベースに格納し、結合処理そのものを減らす処理を行う。その処理により、アクセス処理量が減り、処理効率が上がる。

その処理の操作として「複写」と「併合」の2つの操作が必要になる。その2つの操作を簡単に定義すると「複写」とは、ある関係スキーマに属する一つの関数従属を別の関係スキーマにも重複してもたせる操作であり、「併合」は2つの関係スキーマを一つにまとめる操作である。

しかし「複写」と「併合」の操作をする場合、データの重複によりデータ量が増え、新たな不整合が生じないようにしなければならなく、また重複したデータの更新処理量が増えるという欠点もある。

2. データの分割方法

分散データベースのデータの分割方法には大きく分けて、2つに分かる事ができる。一つは垂直型データ分割という方法であり、表の中でも従属関係を持つ独立性の高い列ごとに表を分割し、これを異なるデータベースサーバで管理する。この分割方法は、データベース自体の独立性と拡張性に富んでいるが、多くの表が必要となる場合、検索に複数の工程を必要とし、ネットワーク上のトラフィックが増大するデメリットがある。

もう一つの方法は水平型データ分割という方法であり、表自体をアクセスされる行集合毎に分割し、各データベースサイトで管理する方法である。これは表自体のフォーマットが同一であるものの、A社とB社、それぞれが頻繁にアクセスする得意先情報表を、双方のデータベースサーバに分割する方法である。

これによりA社のユーザはA社のデータベースサーバ、B社のユーザはB社のデータベースサーバに対して、アクセスすれば、ことが足りるためアクセス頻度が低減できる。つまり、水平型データ分割は、アクセスの分散化においたデータ分割である。しかし、表の列が変更した場合、全てのデータベースサーバの表、全てを変更しなければならない欠点がある。

以上、2つの分割方法があるが、今回の研究では水平データ分割を用いて、研究していきたいと思う。

東海大学大学院理学研究科数学専攻

3. 諸定義

関係データベースの基本概念については，文献 [1] をもとにしており，ここでは本論文で必要なものだけを定義する。

関係スキーマを $< R, F >$ で表す。R は属性集合であり，F は R 上の関数従属の集合である。$< R, F >$ 上の関係 r とは，F に属する全ての関数従属を満たす R 上で定義される関係である。F に属する全ての関数従属を満たす関係を常に満たす関数従属の集合を F^+ と表す。

関係スキーマの系列 $R :< R_1, F_1 >, ..., < R_n, F_n >$ をデータベーススキーマと言う。データベースとは，関係の系列 $D : r_1, r_2, ..., r_n$ で，各 r_i が $< R_i, F_i >$ 上の関係であるものを言う。D に属する関係 r_i を基底関係という。

3.1 複写の定義

データベーススキーマ R に属する二つの関係スキーマ $< R_j, F_j >, < R_k, F_k >$ に対して，$X \subseteq R_j \cap R_k$ かつ $Y \not\subseteq R_j$ を満たす関数従属 $X \rightarrow Y$ が F_k に存在するとする。このとき $X \rightarrow Y$ を $< R_j, F_j >$ に複写するとは，R において，$< R_j, F_j >$ を，$< R_j \cup Y, F_j \cup \{X \rightarrow Y\} >$ で置換する事を複写と呼ぶ。ただし，$< R_k, F_k >$ は変更せずにそのまま残す。

3.2 併合の定義

データベーススキーマ R に属する二つの関係スキーマ $< R_j, F_j >, < R_k, F_k >$ に対して，関数従属 $R_j \cap R_k \rightarrow R_k$ が F_k^+ に存在するとする。このとき，この二つの関係スキーマを，新しい一つの関係スキーマ $< R_j \cap R_k, F_j \cap F_k >$ で置換する操作を併合と定義する。

4. 例

ある大学での履修管理システムを一例とする。このシステムでは，主に学生が持っている学籍番号と，その学生が履修している科目コードが挿入されている {履修} と、科目コードと科目の詳細（科目コード，科目名，担当教員，教室，曜日，時限）が挿入されている {科目} の2つのスキーマが使われているとする。

そこで学生が履修している科目の詳細を検索するとする。その場合、{履修} と {科目} を結合処理をして、検索処理をしなければならない。また学生が他学部の科目（他のサーバに存在するデータ）を履修している場合、そ

の科目のデータが入っているサーバを探す処理をしなければならない。そこで、あらかじめ２つのスキーマを併合して、新しいスキーマ｛履修科目｝を作る。

- ｛履修｝:(学籍番号，科目コード)
- ｛科目｝:(科目コード，科目名，担当教員，教室，曜日，時限)
- ｛履修科目｝:(学籍番号，科目コード，科目名，担当教員，教室，曜日，時限)

※ ｛履修｝の中で挿入されている科目コードは、｛科目｝の中で存在する科目コードでなければならない。

サーバ

大学の各学部に設置する。

データの分割方法

｛履修｝、｛履修科目｝は学生が所属する学部によって，分割し，｛科目｝はその科目が開講される学科によって，分割する。

学生が履修した時の処理は，｛履修｝にデータを追加した時に，｛履修科目｝にもデータを追加する処理を行う。

｛履修科目｝スキーマを作成する事により，学生が履修科目の詳細を検索する場合，結合処理や他のサーバにデータを探す処理が不要になるため，検索処理の効率が向上する。

5. 実験内容

以下の構造のデータベースで実験を行う。

- 「複写」と「併合」を用いた集中型データベース。
- 「複写」と「併合」を用いない集中型データベース。
- 「複写」と「併合」を用いた分散型データベース。
- 「複写」と「併合」を用いない分散型データベース。

実験する内容は，

- それぞれの構造のデータ量。
- それぞれの構造においての処理速度（処理性能）。
  - 結合処理を含んだ検索処理。
  - データの更新処理・削除処理。
  - データの挿入処理。
- 「複写」と「併合」は集中型データベースだけではなく，分散型データベースにおいても効率化できるか。

次に、上記の　と　の構造で実験を行った場合、どのような結果が生じるか予想での結果を表にしてみる。

比較項目	複写・併合を用いた場合	複写・併合を用いない場合
データ量	×	○
結合処理を含んだ検索処理	○	×
削除・更新処理	×	○
挿入処理	△	△

- テーブルを複写と併合をすると、あらかじめ結合したスキーマにより、データの重複部分が生じ、データ量が増える。
- テーブルを複写と併合をすると、あらかじめ結合されているため、処理量が減る。
- テーブルを複写と併合をすると、あらかじめ結合したスキーマにより、データの重複部分が生じ、データの更新・削除処理の処理量が増える。
- 従属したデータなどのデータの重複を気にせずに挿入する場合、複写や併合を用いる場合の方が、結合処理などをする可能性があるため、処理量が多くなるが、従属したデータなどのデータの重複を気にしなければならない場合、データの重複を確認する時に検索処理が必要になるため、処理量の比較が難しい。
  (先述の例で表すと，｛履修｝にデータを挿入する際に，その学生が挿入しようとしている科目が既に履修している科目と時間（曜日・時限）が重なっていないか検索処理を行って，確かめなければならない。その際に，「複写」と「併合」を用いていない場合，｛履修｝と｛科目｝を結合する必要が出てくる。)

6. 最後に

分散型データベースにおいて、データベーススキーマが与えられたとき，検索処理の効率を上げるために，関係スキーマを変形する「複写」と「併合」の二つの操作を定義した。

今後は、上記の実験を行う予定だが，　の実験では，「複写」と「併合」を用いた分散型データベースが，集中型データベースに比べ，処理速度が近くなる事が予想される。よって、検索処理において、分散型データベースでも集中型データベース同様、「複写」と「併合」を用いることにより、効率化できることが予想される。

また　の実験では，「複写」と「併合」を用いた場合の方が，処理速度が遅くなることが予想される。さらに、「複写」と「併合」を用いる事により、データ量が増えることも欠点として、あげられる。

謝辞

本論文を書くにあたり、ご指導をして頂きました東海大学 峯崎俊哉助教授に心から感謝いたします。

参考文献

[1] 中西 通雄, 葛山 善基, 伊藤 実, 橋本 昭洋:"検索処理を高速化するためのデータベーススキーマの設計手法", 電気情報通信学会論文誌 (D-1),J78-D-1,1,pp.44-54(1995)