多次元科学データに対する効率的な問合せのための階層的集約構造

加藤　芳秀[†]　　加藤　　翔[††]　　石川　佳治[†,†††]

† 名古屋大学情報基盤センター
†† 名古屋大学工学部電気電子・情報工学科専攻情報工学コース
††† 国立情報学研究所
E-mail: †yosihide@el.itc.nagoya-u.ac.jp, ††kato@db.itc.nagoya-u.ac.jp, †††ishikawa@itc.nagoya-u.ac.jp

あらまし　科学分野の研究においては，観測により得られた大規模な多次元データに対する関数近似がしばしば行われる．実世界において観測される現象は実際には連続的な値をとるため，観測される離散的なデータに対して曲線フィッティングを行ったり，データに含まれるノイズを除去したりするためである．従来，このような処理はデータベースとは独立したプロセスとして行われてきたが，大規模データを有効に活用するために，このような処理をデータベースに統合して，問合せ処理を効率化することが求められている．本稿では，多次元科学データに対する近似的問合せ処理手法を提案する．本手法は，多次元データ，あるいは多次元データから求められる近似関数に対する効率的な問合せ処理を実現するものである．本手法では，多次元データや近似関数に関する統計量を事前に集約しておき，その情報を階層的な構造を用いて保持する．この構造は一種の索引の役割を果たし，上位の階層は粒度の粗い統計情報を，下位の階層は粒度の詳細な統計情報を保持する．問合せ処理時には，上位の階層から下位方向にこの構造を探索する．必要な情報を段階的に絞り込むことができるため，問合せ処理を効率化できる．

キーワード　近似的問合せ，科学データベース，STING

Efficient Query Processing for Multidimensional Scientific Data Using Hierarchical Aggregation Structure

Yoshihide KATO[†], Sho KATO[††], and Yoshiharu ISHIKAWA[†,†††]

† Information Technology Center, Nagoya University
†† Department of Information Engineering, Nagoya University
††† National Institute on Informatics
E-mail: †yosihide@el.itc.nagoya-u.ac.jp, ††kato@db.itc.nagoya-u.ac.jp, †††ishikawa@itc.nagoya-u.ac.jp

1. はじめに

近年，科学分野の研究において，観測やシミュレーションなどにより得られた大規模なデータを活用することが重要となっている．ジム・グレイが「第 4 のパラダイム」と呼んだように，大規模データを前提としたデータ集約 (data-intensive) 的な科学の研究は，新たなパラダイムとなっている [2]．科学分野のデータを対象として，従来から科学データベース (scientific database) の研究が進められてきたが，このような流れを受けて，さらに大規模なデータの活用のための技術開発が盛んになってきており，データベースの研究分野に対する期待が高まってきている [5].

本稿では，観測あるいはシミュレーションにより得られた大規模な多次元データに対する近似的問合せ処理手法を提案する．本手法では，多次元データに関する統計量を事前に集約し，その情報を階層的なグリッド構造を用いて保持する．上位の階層は粒度の粗い統計情報を，下位の階層は粒度の詳細な統計情報を保持する．問合せ処理時には，上位の階層から下位方向にこの構造を探索する．必要な情報を段階的に絞り込むことができるため，問合せ処理を効率化できる．本手法は，Wang らにより提案された，階層的なグリッド構造により空間的なデータの統計情報を集約する STING [7] のアイデアを拡張したものと位置付けることができる．

本稿の構成は以下のとおりである．次の 2 節では，多次元科学データに対する問合せ処理が実現すべき機能について説明し，従来の手法の問題点について議論する．3 節では，階層的集約構造を用いた近似的問合せ処理を提案する．4 節では，提案した階層的集約構造の有効性を確認するための実験について報告

する．5 節は，本稿のまとめである．

2. 多次元科学データに対する問合せ処理

本節では，まず，具体的な例を用いて，多次元科学データに対する問合せ処理が実現すべき機能について説明する．次に，従来の手法がそれらの機能をどの程度実現し，どのような課題が残されているかを明らかにする．

例として，ある広大な領域に多数のセンサを配置し，温度を観測する状況を考える．観測されるデータはタプルとして表現できる．例えば，時刻 t，位置 (x，y) において温度 temp が観測された場合，(t，x，y，temp) というタプルで表現できる．これらのタプルの集合としてデータベースを定義できるが，このような形式による観測データの表現には，以下のような問題点がある．

- **連続的な値を表現できない**：実世界において観測される現象は，実際には連続的な値をとっている．観測されるデータは離散的であるが，科学データベースを活用するユーザは，観測値が連続的な値をとるものとイメージして，データの分析を行いたい場合がある．観測値のタプル集合による表現は，このような分析には適していない．
- **観測値に存在するノイズに対応できない**：センサ等により観測されるデータには，誤差，ゆらぎ，データ取得のミスなどによるノイズが存在する．ノイズを含んだデータを直に分析するのは，状況によっては必ずしも適切ではない．

上記の問題に対して一般にとられるアプローチは，データベースから分析に必要なデータを問合せにより抽出し，統計処理ソフトウェアを用いて回帰分析による曲線フィッティングを行ったり，ノイズ除去などのクリーニング処理を適用することである．しかしこのようなアプローチでは，データベースシステムが提供する大規模データの効率的処理機能を有効に活用できないため，観測データが大規模の場合，その処理は困難である．

この問題を解決するひとつのアプローチは，統計処理の機能をデータベースと統合することである．このような背景を受け，統計処理の機能を統合したデータベースシステムがいくつか提案されている．以下では，それらのデータベースシステムについて概観し，その問題点を述べる．

2.1 MauveDB

Deshpande と Madden は，線形回帰と線形補間の機能をサポートしたデータベースシステムである MauveDB を開発している [3]．MauveDB では，データベースに格納されている離散的なデータに対して，線形回帰，あるいは線形補間により近似関数を求めることができ，これに対する問合せを可能にしている．例えば，上で述べたような温度観測のデータがデータベースに格納されている場合，そのデータの回帰関数に対して，「温度が 18 度未満である位置についてその座標を求めよ」といった問合せが可能である．問合せ結果は離散化されており，事前に定めたグリッドサイズで値を変化させて関数の値をグリッド状に取得する．

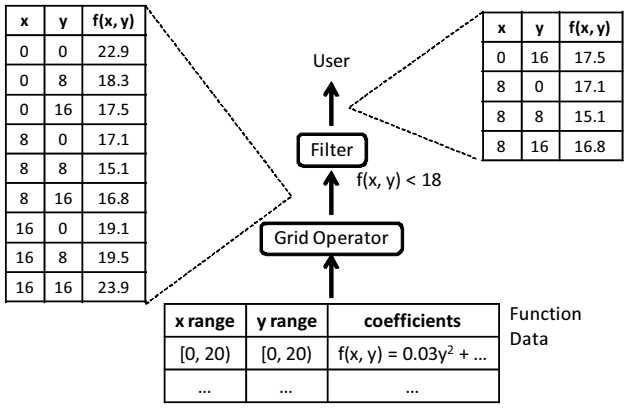

図 1　従来の関数データベースシステムの問合せ処理の流れ

2.2 FunctionDB

Thiagrarajan と Madden は，連続関数値データを表現・操作できるデータベースシステムである FunctionDB を開発している [6]．FunctionDB では，**関数ビュー** (function view) という概念を導入している．これは，データベースに格納される離散的なデータに対して，関数による近似を行った仮想的ビューを提供するもので，ユーザは，データが離散的であることを意識せず，関数により表現されているかのように扱うことができる．FunctionDB でも，MauveDB と同様の問合せが可能であり，問合せ結果は離散化されている．MauveDB とは異なり，グリッドサイズは，問合せ時にユーザが任意に指定することができ，より柔軟な問合せが可能である．

2.3 従来のデータベースシステムの問題点

これらのデータベースシステムに共通するのは，問合せ処理時に関数値をグリッド状に評価しなければならない点である．フィルタ演算に先立って関数値をグリッド状に評価するため，問合せ条件を満たさないような格子点についても関数値の評価が必要となり，問合せ処理における大きなコスト要因となる．特に，問合せの対象となる関数の計算コストが高い場合，この問題はより顕著となる．

問題点を明らかにするために，従来のデータベースシステムの問合せ処理の流れを図 1 に示す．MauveDB では，データ追加時に近似関数を効率的に再計算するために，関数に関する情報は中間的な表現形式でデータベース内に保持されている．問合せ処理時には，中間表現は，図の最も下にあるような関数データに変換される．続いて，グリッド状に関数値をサンプリングし，その結果をフィルタ演算で処理する．関数値のサンプリングは領域全体にわたって行う必要があるため，問合せ処理は非効率的である．

FunctionDB でも，問合せ処理の時点でグリッド処理を実行する．FunctionDB では，関数データはデータベース内に関数テーブルとして格納されている．問合せ処理時には，関数テーブル中の関数に対して，グリッド処理によりグリッド格子点における関数値を計算し，その結果をフィルタ演算で処理して，最終的な結果を求める．FunctionDB では，いくつかの代数的手法を用いてグリッド処理の対象を絞りこんでいるものの，依

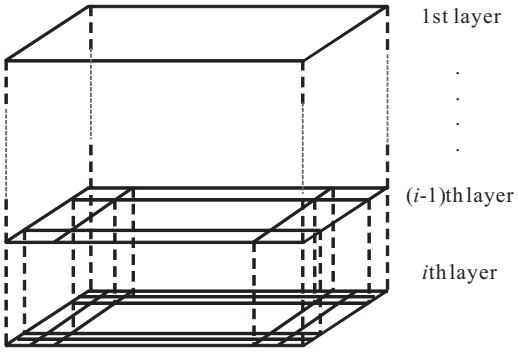

図 2　STING の階層構造

然として問合せ処理の時点でグリッド処理を実行しなければならず，問合せ処理時の作業が多いことには変わりはない．特に，関数が多項式のような単純な形式で記述できない場合，代数的処理にも限界があり，問合せ最適化の効果も限定的とならざるをえない．

3.　階層的集約構造に基づく近似的な問合せ処理

2 節で述べた問題に対処するために，本節では，階層的集約構造を用いた近似的問合せ処理を提案する．本手法は，階層的なグリッド構造により空間的なデータの統計情報を集約する STING [7] をヒントにしている．MauveDB や FunctionDB では問合せ処理時にグリッド処理を行っていたが，本手法では，事前にグリッド処理を実行し，その結果に基づき索引を構築する．これにより問合せ処理の効率向上が期待できる．以下では，本手法で使用する階層的集約構造を**集約索引** (aggregation index) と省略して呼ぶことにする．まず，STING を概観し，続いて本手法について説明する．

3.1　STING

Wang らにより提案された STING (STatistical INformation Grid-based method) [7] は，空間データマイニングの一手法である．階層的なグリッド分割がキーとなるアイデアである．

図 2 に，STING が用いる階層構造のイメージを示す．階層構造は複数のレイヤから構成され，一つのレイヤが空間全体（図においては 2 次元空間）に対応している．各レイヤは複数のグリッドセルに分割される．レイヤごとに分割の粒度は異なり，最上位のレイヤは一つのセルのみから構成される．その他のレイヤは，一つ上のレイヤのセルをより細かいセルに分割したものとなっている．図中では，レイヤは 2 次元空間を表現し，各セルはレイヤが低くなるごとに 4 分割されているが，原理的には，空間の次元数と分割数は任意に定めることができる．

レイヤ中の各セルは，セル内に含まれるデータ点に関する統計情報を格納する．格納する情報は，データ数や属性値の平均，標準偏差，最大値，最小値などの統計量，及びそのグリッドセルにおいて，属性値が従う分布の種類である．例えば，各位置に対する温度の測定値 (`x`, `y`, `temp`) が対象データの場合を考えると，空間属性である x, y の値により，データがどのグリッドセルに属するのかが定まり，グリッドセルごとに，それに属するデータの数，及び温度の平均，標準偏差，最大値，最小値を格納する．また，温度がそのグリッドセルにおいてどのような分布（例えば，ガウス分布）に従うかが管理される．統計情報は，最下位のレイヤについては，データ点から直に計算する．それ以外のレイヤについては，一つ下のレイヤの統計情報をもとに計算する．すなわち，$(i-1)$ 番目のレイヤのセルの情報は，i 番目のレイヤの対応する 4 つのセルから求められる．$n_A(c)$, $m_A(c)$, $s_A(c)$, $min_A(c)$, $max_A(c)$ をそれぞれ，属性 A に関してセル c が保持するデータの数，平均，標準偏差，最小値，最大値とすると，これらの値は，セル c に対応する下位レイヤのセルの集合 $children(c)$ の統計情報から以下のように計算できる．

$$n_A(c) = \sum_{c' \in children(c)} n_A(c')$$

$$m_A(c) = \frac{\sum_{c' \in children(c)} m_A(c') n_A(c')}{n_A(c)}$$

$$s_A(c) = \sqrt{\frac{\sum_{c' \in children(c)} \{s_A(c')^2 + m_A(c')^2\} n_A(c')}{n_A(c)} - m_A(c)^2}$$

$$min_A(c) = \min_{c' \in children(c')} min_A(c')$$

$$max_A(c) = \max_{c' \in children(c')} max_A(c')$$

[7] では，上述のデータ構造を対象とした問合せ言語を提供している．例えば，「温度が $(15, 20)$ の範囲にあるような確率が 80%以上であるような領域を求めよ」というような問合せが記述できる．問合せ処理は，最上位レイヤのセルから開始される．階層構造を下位方向に向けて探索し，条件を満たす最下位レイヤのセルを求め，統計量を用いた推定処理が行われる．元データにアクセスすることなく問合せを処理することにより，効率的な問合せ処理を実現している．

3.2　提案手法の基本的なアイデア

提案手法は，STING で用いられる階層構造をベースとするが，その基本的なアイデアは次の 2 点にまとめられる．

（1）　集約索引では，STING と同様に階層的に統計情報を集約する．観測データを対象に集約索引を構築する場合は，STING と同様の方法により集約索引を構築する．連続関数データを対象とする場合は，関数の値をグリッド状にサンプリングし，サンプリングした点を対象として集約索引を構築する．

（2）　問合せが与えられると，集約索引に保存された統計情報を用いて，処理の効率化を図る．問合せ処理時には，集約索引を上位から下位方向に向けて探索することにより，必要な情報を段階的に絞り込みながら，問合せ要求に答える．

このアプローチの問題点は，索引構築時にグリッドサイズを決めなければならないため，ユーザが問合せ時にグリッドサイズを動的に与えられない点である．これは問合せ処理の効率とのトレードオフであり，処理速度が優先される場面，例えば，対話的に問合せを行うような場面では，ユーザにとってメリットが大きいと考えられる．

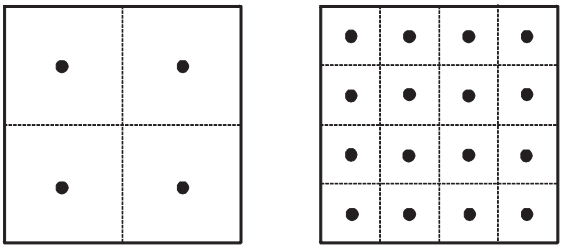

図 3　階層的な空間分割

3.3　集約索引の構築

集約索引の構築時には，階層の深さ，分割数，及び，いずれの属性が空間属性であるかを指定する．n 個の属性が空間属性に指定された場合，それらの属性が構成する n 次元空間に対する集約索引を構築する．その他の属性に対しては，統計情報を集約する．例えば，位置 (x, y) における温度 temp に関する集約索引を構築する場合，空間属性として x, y を指定すれば，2 次元空間をレイヤとするような集約索引が構築され，各グリッドセルには，温度に関する統計情報が集約される．

観測データ（離散データ）に対しては，STING と同様の手順で集約索引を構築する．連続関数データに対しては，次節で述べる階層的なサンプリングにより得られたデータ点に対して集約索引を構築する．

3.3.1　階層的なサンプリング

連続関数データに対する集約索引の構築時には，連続関数データに対する階層的なサンプリングを行う．階層的なサンプリングにおいては，STING のように，階層的なレベルを考える．ルートをレベル 0 とし，以下，レベル 1，レベル 2 とレベル番号が増加する．図 3 に示すように，各レベル i においては，空間を $2^i \times 2^i = 4^i$ 個に分割し，各セルの中点における関数値をサンプリング対象とする．分割レベルの最大値は，連続関数に対する集約索引を構築する際，ユーザが直接指定する（このレベルは，集約索引のレベルと同一でなくともよい．その理由については後述する.）．サンプリングにより得られたデータ点に対して集約索引を構築する方法は，STING と同様である．

集約索引では，統計情報に加えて，サンプリング点における関数値も合わせて保持する．関数に対する問合せ結果として利用するためである．各レイヤのセルは，同一レベルの階層のサンプリング点における関数値を保持する．すなわち，集約索引は，対象の関数に対する一種の**実体化ビュー** (materialized view) も含んでいるということができる．ここで注意すべきポイントは，すべてのサンプリング点が集約索引に保持されるわけではないことである．集約索引とサンプリングのレベルは異なっていてもよく，集約索引より深いレベルのサンプリング点は，集約索引には保持されず，統計情報の計算のみに使用する．例えば，2 次元空間において，集約索引のレベルを 6，サンプリングのレベルを 8 とした場合，$2^8 \times 2^8 = 65,536$ 個のサンプリング点が得られる．集約索引の最下位レイヤは，$2^6 \times 2^6 = 4,096$ 個のセルから構成され，近接する 16 個のサンプリング点を 1 グループとして，サンプリング点における関数値の平均値，最大値，最小値などの統計量をとり，集約索引の最下位レイヤの各セルに対するエントリを構築する．

集約索引構築時の 2 つのレベル指定の役割をまとめると，以下のようになる．

- サンプリングのレベルは，集約索引の詳細度を定める．集約索引の統計情報は，指定されたレベルのサンプリング点の関数値に基づき計算されるため，サンプリングのレベルが深いほど，より正確な統計情報が集約索引に格納される．
- 集約索引のレイヤのレベルは，どのレベルの問合せまでを高速に処理するかを定める．レイヤのレベルまでは，サンプリングされた関数値が集約索引に収められている．そのため，このレベルまでの問合せに対しては，新たにサンプリングを行う必要はなく，集約索引のみを利用して高速に必要な情報を得ることが可能となる．逆に，より詳細なレベルに対する問合せについては，統計情報を用いて大まかなフィルタリングはできるものの，最終的な関数値については，関数の情報を用いて動的に求めなくてはならない．

3.4　集約索引を用いた問合せ処理

集約索引を用いた問合せ処理は，最上位レイヤのセルから下位方向に集約索引を探索することを基本とする．問合せ処理に関する具体的なイメージをつかむために，3 つの属性 (x ,y, temp) からなる観測データの関数ビューを対象とした，次のような問合せ処理を考える．

- temp が 18 度未満である位置についてその座標 (x, y) を求めよ．ただし，グリッドレベルは 4 とする．

まず，集約索引の最上位レイヤのセルの統計情報にアクセスする．仮に，このセルが保持する temp の統計情報において，その最小値が 18 度以上であったとすると，そのセルが覆う領域においては，どのサンプリング点も「temp が 18 度未満」という条件を満たさないことが明らかである．したがって，このセルに対応する下位レイヤのセルに対する問合せ処理は不要とわかる．一方，temp の最大値が 18 度以下であったとすると，すべてのサンプリング点が,「temp が 18 度未満」という条件を満たすので，このセルが覆う領域のサンプリング点を求めればよい．それ以外の場合，すなわち temp の最小値が 18 度以下かつ最大値が 18 度以上の場合は，そのセルに対応するレベル 1 の 4 つのセルに対して，同様に処理を再帰的に適用する．この処理は，ユーザが指定したグリッドレベル（すなわち，この例の場合はレベル 4）に到達するまで繰り返される．最終的に残されたセルからサンプリング点を入手し，問合せ処理が終了する．

問合せ処理木のイメージを図 4 に示す．集約索引を用いた問合せ処理は，次の 2 つの処理から構成される．

（1） 指定された条件を満たすサンプリング点から構成されるようなセルを選択する．

（2） 選択されたセルから，サンプリング点，属性値の平均，最大値，最小値などを求める．

ここでのポイントは，効率的な問合せ処理を実現するために，集約索引では，サンプリング点や観測データにアクセスすることなくセルを選択する点である．選択演算の擬似コードを図 5 に示す．*cond* はサンプリング点が満たすべき条件，l は選択される

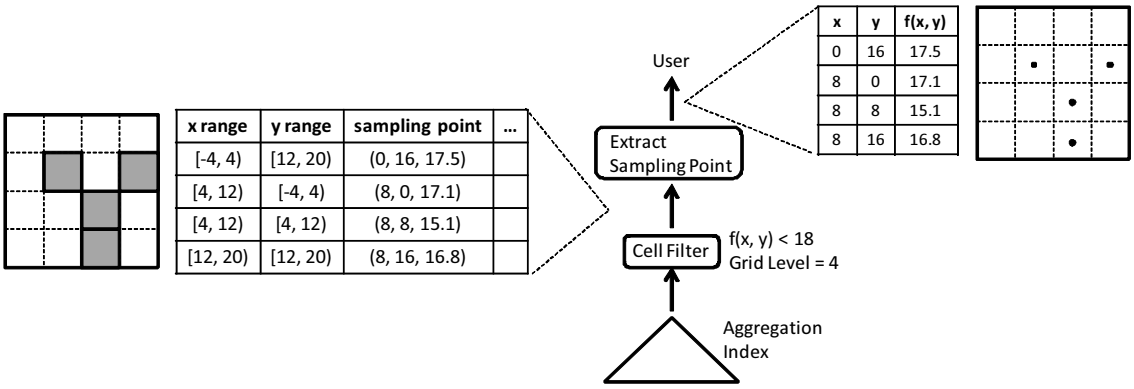

図 4　問合せ処理木のイメージ

入力：集約索引 $Index$，条件 $cond$，グリッドレベル l

```
1  C_0 ← {Index の最上位レイヤのセル }
2  Result ← {}
3  for i from 0 to l do
4     C_{i+1} ← {}
5     for each c ∈ C_i do
6        if satisfy(c, cond) = 1 then
7           Result ← Result ∪ descendant(c, l)
8        else if satisfy(c, cond) = 0.5 then
9           if i < l then
10             C_{i+1} ← C_{i+1} ∪ children(c)
11 return Result
```

図 5　選 択 演 算

セルのグリッドレベルである．$descendant(c, l)$ は，セル c に対応するグリッドレベル l のセルの集合である．$satisfy(c, cond)$ は，セル c 中のすべてのサンプリング点が条件 $cond$ を満たすとき 1 を，いずれのサンプリング点も条件を満たさないとき 0 を，どちらの場合でもないときには 0.5 を返すような関数である．比較演算子を用いた条件や，それらを論理演算子で結合した条件について，$satisfy(c, cond)$ は以下のように定義できる．

- $cond$ が $A > \theta$（A は属性，θ は定数）とき，

$$satisfy(c, cond) = \begin{cases} 1 & min_A(c) > \theta \\ 0.5 & max_A(c) > \theta \geqq min_A(c) \\ 0 & \theta \geqq max_A(c) \end{cases}$$

$cond$ が $A < \theta$ のときも，ほぼ同様に定義できる．

- $cond$ が $cond_1 \wedge cond_2 \wedge \cdots \wedge cond_n$ のとき，

$$satisfy(c, cond) = \min_i satisfy(c, cond_i)$$

- $cond$ が $cond_1 \vee cond_2 \vee \cdots \vee cond_n$ のとき，

$$satisfy(c, cond) = \max_i satisfy(c, cond_i)$$

- $cond$ が $\neg cond'$ のとき，

$$satisfy(c, cond) = 1 - satisfy(c, cond')$$

セルが選択された後の処理は比較的単純である．サンプリング点を求める場合は，選択されたセルに格納されているサンプリング点を抽出すればよい．サンプリング点の集合はテーブルとみなすことができるため，これに対して様々な演算を適用することが可能である．

単に属性値の平均，最大値，最小値などを求める場合には，セルに格納されている統計情報のみを用いて，これらを求めることができる．サンプリング点にアクセスする必要はない．すなわち，$Result$ 中のセルに格納される統計情報から次のようにして，求めることができる．

$$n_A = \sum_{c' \in Result} n_A(c')$$

平均：$m_A = \dfrac{\sum_{c' \in Result} m_A(c') n_A(c')}{n_A}$

最大値：$max_A = \max_{c' \in Result} max_A(c')$

最小値：$min_A = \min_{c' \in Result} min_A(c')$

さらに，図 5 の選択演算の 7 行目を次のように変更すると，より効率的である．

$$Result \leftarrow Result \cup \{c\}$$

選択演算において，$descendant(c, l)$ を $Result$ に追加するのは，c が覆う領域のレベル l のサンプリング点を得るためである．統計情報のみを利用するような問合せにおいては，$descendant(c, l)$ の中のセルの統計情報は，既にセル c に集約されているため，これを利用した方がより効率的である．

最後に，集約索引のグリッドレベルを超える問合せ処理について考える．この場合，集約索引の中には問合せ処理に必要な情報がすべて収められているわけではないため，動的なサンプリングが必要になる．図 6 がその問合せ処理木である．問合せ処理は，次のようになる．まず，集約索引のレベルまでの情報は正確に把握できるため，集約索引の情報を用いて，サンプリングをどの領域で行えばよいかが詳細化できる（この領域は，最下位レイヤのセルの集合として規定される.）．つまり，次のステップであるグリッド演算によるサンプリングの処理コストを抑えることが可能となる．さらに，最終段階におけるフィルタリング処理においても，候補数が削減されているため，効率的に処理できる．

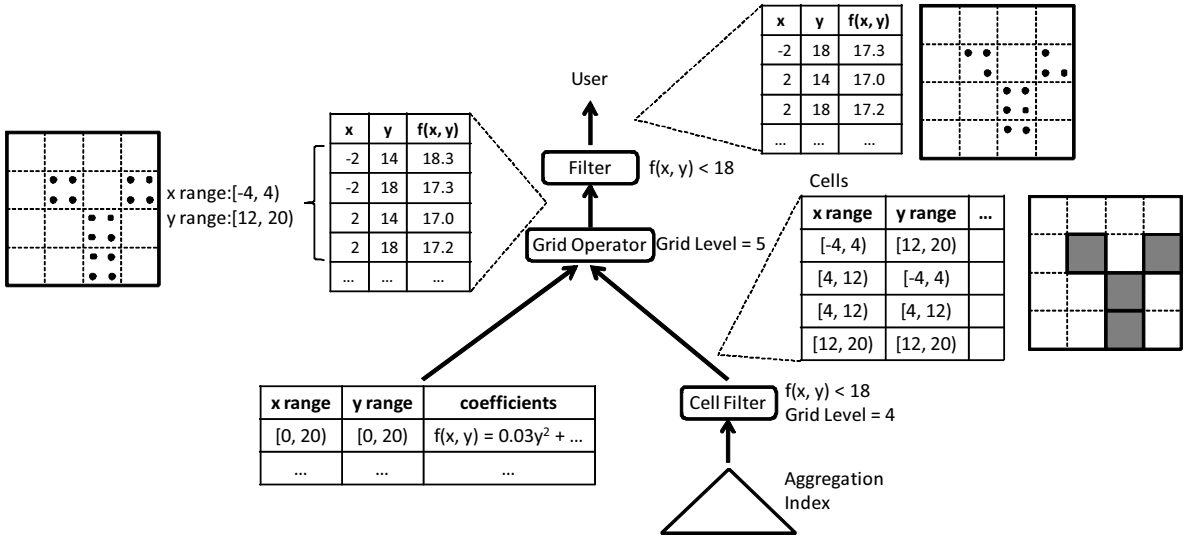

図 6　動的なサンプリング処理

4. 実　　験

提案手法の有効性を確認するための実験を行った．集約索引の構築，及びそれを用いた問合せ処理を実現するプログラムを ruby を用いて実装した．対象となるデータとして,「西スマトラレーダー・雨量計統合降水量データ 第 1.1 版 (WeSRI 1.1)」[4] を使用した(注1)．WeSRI 1.1 は，緯度，経度，時刻，降水量についてのデータである．緯度方向，経度方向に 60 個ずつの計 3600 ヶ所（実際には欠損があり，1517 ヶ所）における降水量のデータである．時間解像度は 30 分である．本実験では，一日分のデータ（2006 年 10 月 28 日のデータ）に対して，集約索引を構築した．降水量は，回帰木による関数近似を行い，緯度，経度，時刻の関数として定義した．集約索引は，緯度，経度，時刻を空間属性として構築した．サンプリングのレベルを 6，レイヤのレベルは 5 とした．回帰木の構築には，GUIDE [1] を使用した．いくつかの問合せについてその動作を調べた．

実験に使用した問合せを以下に挙げる．なお，理解を容易にするために，仮想的な問合せ記述言語による問合せも併記するが，実際には，問合せ処理木を人手により作成して処理を実行している．index を構築した集約索引とし，x，y，t，p はそれぞれ，緯度，経度，時刻，降水量を表すものとする．

問合せ 1

降水量が 20mm/h 以上である地点の座標と時刻を求めよ．

```
SELECT x, y, t FROM index
WHERE p > 20
```

問合せ 2

2006 年 10 月 28 日の午前 6 時から午後 6 時のうち，降水量が最も多かった地点の座標と時刻を求めよ．

```
SELECT x, y, t, MAX(p) FROM index
WHERE "2006/10/28 06:00" < t < "2006/10/28 18:00"
```

問合せ 3

経度が [99.8758, 100.2281]，緯度が [-1.0828, -0.6188] の領域で，1mm/h 以上，2mm/h 未満の降水量を記録した地点の座標を求めよ．

```
SELECT x, y FROM index
WEHRE 1 < p < 2
AND 99.8758 < x < 100.2281
AND -1.0828 < y < -0.6188
```

表 1　問合せ処理時間の比較 (グリッドレベル 5)

	処理時間 (s)		
	問合せ 1	問合せ 2	問合せ 3
集約索引使用	0.84	0.77	0.11
集約索引未使用	0.83	0.81	0.89

問合せのグリッドレベルは，5 もしくは 6 に設定し，問合せ処理を実行した．集約索引のレベルは 5 であるため，問合せのグリッドレベルが 5 のとき，サンプリング点の関数値は集約索引から獲得される．問合せのグリッドレベルが 6 のときは，動的サンプリングにより関数値を関数から直接計算することになる．

集約索引を用いることにより，問合せ処理の効率がどの程度改善されたかを評価するために，提案手法による問合せ処理と，次のようなサンプリングに基づく問合せ処理を実行し，その処理時間を比較した．

- 問合せで指定したグリッドレベルで全領域にわたってサンプリングを行い，各サンプリング点における関数値を求めたのち，問合せ条件を満たすサンプリング点を抽出する．

各問合せに対する処理時間を表 1 及び表 2 に示す．グリッドレベルが小さいときは，そもそも，サンプリングすべき点自体

(注1)：WeSRI 1.1 は文部科学省の HARIMAU (Hydrometeorological Array for ISV-Monsoon Automonitoring) による観測と DIAS (Data Integration and Analysis System) によるデータ作成技術を用いて作成された．

表 2　問合せ処理時間の比較 (グリッドレベル 6)

	処理時間 (s)		
	問合せ 1	問合せ 2	問合せ 3
集約索引使用	0.91	2.45	0.30
集約索引未使用	7.53	7.38	8.00

が少ないためそのオーバーヘッドの影響も小さく，表 1 に示すように，集約索引を用いても問合せ処理の効率はわずかしか向上しない．一方，グリッドレベルが大きいとき，集約索引を用いた問合せ処理においても動的サンプリングを行う必要が生じるが，領域を絞り込んだうえでサンプリングできるため，単純なサンプリングに基づく問合せ処理と比べて効率が大幅に向上している．この結果から，本稿で提案した集約索引を用いることにより，関数データに対する問合せ処理の効率が向上することが確認できた．

5. おわりに

本稿では，連続関数データに対する問合せ処理を効率化するための索引構造を提案した．STING に似た階層的な情報集約のためのデータ構造を活用することにより，コストの高い関数からの動的サンプリング処理のオーバーヘッドを抑えることができる．本稿では，単純な条件による問合せと集約演算処理の方法を示したが，今後は他の種類の問合せ，例えば，`GROUP BY` 句を用いるような問合せに対しても一般化を図りたいと考えている．また，FunctionDB では不等式制約を用いて問合せ条件を記述できるが，そのような問合せについても今後検討したい．また，現状では，ユーザインタフェースが整っておらず，このままではユーザが利用するのは困難である．ユーザビリティを高めるために，集約索引を利用するための問合せ言語を設計する必要がある．今回の実験では，データの規模は比較的小規模であったので，大規模なデータを対象とした実験についても今後実施したい．

謝　　辞

本研究の一部は,「データ統合・解析システム（DIAS）」プロジェクト経費による．

文　　献

[1] Leo Breiman, Jerome H. Freidman, Richard A. Olshen, and Charles J. Stone. Classification and Regression Trees. Wadsworth International Group, 1984.

[2] Tony Hey, Stewart Tansley, and Kristin Tolle, editors. *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research, 2009.

[3] Amol Deshpande and Samuel Maden. MauveDB: Supporting model-based user views in database systems. In *Proc. ACM SIGMOD*, 2006.

[4] 上米良 秀行．西スマトラレーダー・雨量計統合降水量データ 第 1.1 版．データ統合・解析システム，2009.

[5] Arie Shoshani and Doron Rotem, editors. *Scientific Data Management: Challenges, Technology, and Deployment*. CRC Press, 2009.

[6] Arvind Thiagarajan and Samuel Madden. Querying continuous functions in a database system. In *Proc. ACM SIGMOD*, pp. 791–804, 2008.

[7] Wei Wang, Jiong Yang, and Richard R. Muntz. STING: A statistical information grid approach to spatial data mining. In *Proc. VLDB*, pp. 186–195, 1997.