HDF-EOS のための Bitmap 索引を用いた問合せの実現
－パーソナルユースの科学データ管理システムの開発に向けて－

植村 亜以† 渡辺 知恵美‡ 城 和貴†

†奈良女子大学大学院人間文化研究科 〒630-8506 奈良県奈良市北魚屋西町
‡お茶の水女子大学理学部情報科学科 〒112-8610 東京都文京区大塚 2-1-1
E-mail: †(poo, joe)@ics.nara-wu.ac.jp , ‡chiemi@acm.org

あらまし 近年，地球観測機器の高機能化,コンピュータの高性能化により,シミュレーションや分析に使用するデータは個人でも数 G,TB レベルに達しており，個人レベルでのデータ管理の需要が高まっている．しかしながら DBMS を個人利用する場合，学習コスト，可搬性，柔軟性がネックになり，あまり利用されていないのが現状である．そこで,我々は科学データの統一フォーマットとして注目されている HDF ファイルをベースに,科学者のために必要最低限の機能を備えた簡易データ管理システムを開発している．本稿では，特に地球大気科学分野に焦点を当て，HDF フォーマットの地球科学用サブセットとして NASA を中心に普及している HDF-EOS データを対象とし，簡易的な問合せのための Bitmap 索引構造を HDF-EOS ファイルに埋め込み,拡張ライブラリとして提供する．

キーワード 科学 DB，時空間 DB，多次元 DB

The Realization of Inquiry Used by Bitmap Index for HDF-EOS
-The Development of Scientific Data Management System for Personal-

Ai UEMURA† , Chiemi WATANABE‡ and Kazuki JOE†

† Graduate School of Humanity and Science, Nara Women's University Kitauoya nishi-machi,Nara,630-8506, Japan

‡ Faculty of Science, Ochanomizu University 2-1-1,Otsuka,Bunkyo-ku,Tokyo 112-8610, Japan
E-mail: †(poo, joe)@ics.nara-wu.ac.jp , ‡chiemi@acm.org

Abstract Nowadays, data management systems for massive scientific data are becoming more and more important due to the rapid development of scientific instruments and computer simulation technologies. However, it seems that scientists tend to avoid using conventional database systems, mainly from the following reasons: 1) learning the usage of database tools usually takes long time; and 2) once scientific data being analyzed are loaded to a database system, one can no longer manipulate the data using his/her application programs in a direct manner. On the other hand, HDF (Hierarchical Data Format), which is one of universal formats for scientific data, provides rich functionalities for managing multiple large datasets and their meta-data. However, HDF does not provide query functions, such as non-procedural query language, automatic parallelism, and sophisticated index structures.

From the above mentioned observations, we propose a novel data management system based on HDF-EOS formatted file, by extending functions necessary for managing and querying scientific data efficiently. As an initial step of our research project, we have developed query functionalities powered by the WAH compressed bitmap index structure, which can be embedded in the HDF-EOS file. We have conducted an experimental evaluation, and observed that the query performance is three times faster than that without index.

Keyword Scientific Database, Spatio-Temporal Database, Multi-Dimensional Database

1. はじめに

地球大気科学分野におけるデータベース利用が近年注目を集めている．例えば地球観測のための測定機器の高機能化,コンピュータの高性能化による計算能力の著しい発達などにより,シミュレーションや分析に使用するデータはますます肥大化している．

NASA などは数百 TB 若しくは数 PB にも及ぶ地球観測データを管理しており,世界中の研究者は Web を通してそれらのデータをダウンロードできるようになっている.それゆえ研究者個人で利用できるデータも飛躍的に増大し,個人レベルでも数百 GB,数 TB のデータを分析対象として網羅的に調べることが可能となってきた．そのような大量のデータを個人で管理することは困難であり,DBMS 導入の需要も少なくないと考えられる.しかし現在，科学者はデータベースシステムの利用を避ける傾向にある[1][2]．文献[2]における分析では，その理由として以下の点が挙げられている．

- **学習コストがかかる:**
  スキーマ定義やデータの格納，SQL 問合せ，プログラミング言語への埋め込み，アクセス権の設定等データを管理するために一から学習しなければならないことが多

数あり，データ管理が専門でない科学者にとっては学習コストが障害となる．

- **インピーダンスミスマッチの問題：**
  単純な多次元配列から表形式に変換が必要となり，またデータベースからデータを取り出した後も，再び多次元配列に入れなおすなどの手間がかかる．
- **既存のアプリケーションから使えなくなる：**
  科学者が使い慣れている分析ツールや可視化ツールが DBMS 対応でない場合，これまでの様に分析ツールから利用することが難しくなる．またDBMS対応である場合もファイル形式のときとは異なる利用方法を要求されるため，それについても新たに学習しなければならない．
- **DBMS 対応のアプリケーションが要求される：**
  DBMS は既存の分析ツールが持つような多様なアクセスや分析方法，可視化機能を持ち合わせているわけではなく，データを格納した DBMS に対応したアプリケーションも必要となる．
- **可搬性,柔軟性が失われる：**
  一度データベース化してしまうと，ファイルのように気軽に他の計算機や他人にデータを移動させることが困難になる．

逆に科学者がデータベースシステムに期待するのは複数の大量なデータに対する高機能かつ高速な検索機能である．つまり巨大なデータの中から範囲を指定して部分データを取得するだけでなく，データ値を条件にした検索や複数のデータセットにまたがる検索が求められている．特に近年では衛星機器の高機能化により一つの衛星に多種類のセンサを搭載して多様なデータを同時に取得することが可能であり，それらのデータの相関関係を分析したいという要求が高まっている．そのような場合に，複数のデータセットの結合なども必要となる．また 1 つのデータセットの容量が数 GB と巨大になった場合には，必要な部分データセットだけを高速にメモリ上にロードするための索引等も必要である．

一方，地球観測分野で扱われているファイルは，従来データの提供元や利用目的によって様々な形式を各々提供してきたため，NetCDF，FITS など非常に多数のデータ形式が存在していたが，近年ではデータ形式を統一化しようという意識が高まっている．HDF (Hierarchical Data Format)[3]は地球観測データの統一ファイルフォーマットとして NASA 等ですでに採用されている形式である．HDF ではフォーマットを定義するとともに C および FORTRAN 用の API を提供している．階層的に複数の種類の異なる複数のデータセットを格納し，データセット毎にメタデータを定義できるなど，データ配布だけでなく管理まで想定した豊富な機能を備えている．また，HDF はデータセット自体を BLOB(Binary Large OBject)として扱うが，地球観測データ用拡張フォーマットである HDF-EOS[4]形式によって，POINT，SWATH,GRID という 3 種類のデータ表現形式が定義されており，それらのデータ構造を生かした部分データセットへの柔軟なアクセスが可能である．

しかしながら現段階では HDF-EOS データにおいてデータ構造を生かした高速かつ高機能な検索関数は提供されていない．特に，巨大なデータセットから部分セットを取り出す場合，データ値による検索を行うための索引生成機能は十分に需要が高い．

そこで我々は地球大気科学者の個人利用のデータ管理システムとして，従来の RDBMS を利用するのではなく,HDF-EOS ファイルに対して個人利用のデータ管理に最低限必要な機能を備えるというアプローチで簡易データ管理システムを提供する．

その初期段階として，我々はデータモデルのための基礎的な検索機能である選択関数の実装を行った．現在，HDF-EOS 用の API では部分格子の抽出は提供されているものの，例えば「180<temperature<220」というようなデータ値を条件とした単純な選択関数は提供されていない．このような条件でデータを絞り込む場合は，(部分)格子を一度メモリ上に取り出した後，配列の各要素を順次確認しなければならず，巨大なデータから条件に合った部分格子だけを取り出すのに相当な時間と労力が必要であった．そこで我々は HDF-EOS ファイルに高圧縮なビットマップ索引を埋め込み，単純な条件検索を高速に行うための API 拡張ライブラリを提供した．

本稿では，地球科学者の個人利用のための観測データ管理の実際について焦点を当てて分析し，その上で HDF-EOS ファイルをベースにした簡易データ管理システムを提案する．さらに提案システムを実装するための初期段階として，データ値を条件とした単純な選択関数を高速に行うための関数を実装する．

以下，第2節で科学者の個人利用において必要とされる機能について考察したのち，それを踏まえたデータ管理システムを提案する．また第 3 節にて我々が提案システムのベースとするファイルフォーマットである HDF，HDF-EOS について説明し,第4節にて今回実装している機能について述べ，第5節にて実装関数の性能の検証行う．

2. 地球科学者のためのデータ管理システム

2.1. 個人利用観測データ管理の実際

本節では科学者がデータ管理システムに求める機能として何を期待しているかについて考察する．

まずデータベースの機能の中から科学者が必要とする機能およびそうでない機能についてデータの個人利用に焦点を絞って考察したものを表1に示す．トランザクション処理,同時実行制御，アクセス制御などの複数ユーザの利用を想定した機能はあまり利用されない．また観測データは更新がほとんどないため高性能なリカバリ機能も特に必要ではない．一方，科学者がデータベースシステムに期待するのは複数

の大量なデータに対する高機能かつ高速な検索であり，SQL のような非手続き的言語やビュー機能などが待望されている．

必要とされる機能	あまり利用しない機能
問合せ機能	トランザクション管理
アクセス最適化	リカバリ機能
ビュー機能	同時実行制御

表1：DBMS の基本機能および科学者が求める機能

次に，科学者がデータ管理システムに求める条件について第 1 節にあげた DBMS 利用における問題点を元に考察すると，以下の点が重要となる．

- **学習コストの軽減：**
  導入および利用のために科学者が新たに学習しなければならない項目ができるだけ発生しないようにする必要がある．
- **既存のアプリケーションでの利用：**
  管理されているファイルや検索結果データを既存の分析ツールや可視化ツールなどで利用できるようにする必要がある．
- **可搬性の維持：**
  ファイル感覚で気軽にデータを受け渡しできるような可搬性を持ち合わせる必要がある．

2.2. システム概要

2.1 節におけるような考察に基づき，我々は地球大気科学者のためのデータ管理システムを提供するために，従来の RDBMS を利用するのではなく,HDF-EOS ファイルに対して個人利用のデータ管理に必要な機能を備えるというアプローチを採用することとした．

システム構成を図1に示す．まず HDF-EOS の拡張ライブラリとして EOSDB（図 1①）ライブラリを提供することによってデータ管理・問合せのための機能を関数として提供する．我々は上記の分析から以下の機能を提供する．

1．問合せ機能の提供

HDF ライブラリが提供する階層的なデータセット管理構造やメタデータ構造，および HDF-EOS が提供する地球観測用データ構造を利用した問合せ機能を C 言語および FORTRAN 言語用関数として提供する．具体的にどのような問合せを提供するかについては第 4 節にて述べる．

2．索引を利用した問合せ最適化機能

データセットに対する問合せを高速に行うための索引構造を HDF ファイル内部に埋め込み，これを利用した問合せ最適化機能を提供する．

3．ビューデータセット（図中②）

DBMS のビュー表と同様に，HDF で階層的に管理されるデータセットの中に，問合せ関数を利用した仮想的なデータセットを提供する．

4．仮想データセットファイル機能（図中③）

HDF-EOS 内に格納されている1データセット（もしくはビューデータセット）を仮想的な1ファイルとして外部アプリケーションからアクセスするための機能を提供する．

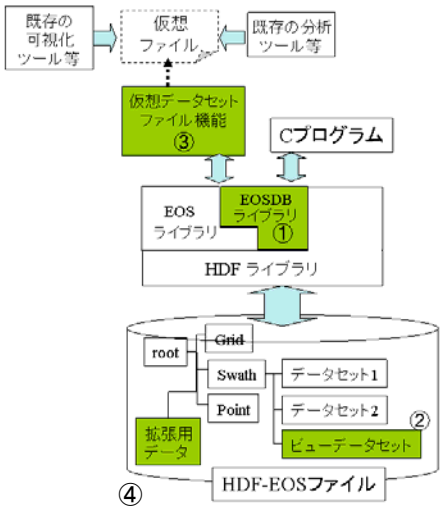


図１　システム構成

なお，これらの機能で必要な内部データ（例えば索引データやそのメタ情報など）は，HDF ライブラリを用いて格納する（図１④）．これらのデータはその上位ライブラリである HDF-EOS ライブラリから隠蔽され，通常の HDF-EOS ライブラリとしても正常に利用できる．

本システムの特徴は以下の点である．

学習コスト：

HDF-EOS のデータモデルを利用した問合せ関数を HDF-EOS の拡張関数として提供しているため，新たな言語等を覚える必要はなく，我々が提供する新たな関数を数個覚えるだけでよい．

既存のアプリケーションでの利用：

通常の HDF-EOS ファイルとしても利用できるほか，ビューデータセットや仮想データセットファイルを利用することにより問合せの結果を既存のアプリケーションで扱うことができる．

可搬性：

HDF－EOS 内にデータベース機能を追加するという形を取っているため,元々のファイルの可搬性が損なわれることはない．

これにより,第2項で考察した科学者の要望がある程度反映されたデータ管理システムが実現するものと考える．なお，本稿では上記の機能のうち1および2の一部を提供する．

3. HDF および HDF-EOS のデータ構造

本稿にて取り上げている HDF および HDF-EOS について述べたのち，提供すべき問合せ機能について述べる．

3.1. HDF

HDF とは Hierarchical Data Format（階層型データフォーマット）を略したもので，イリノイ大学の National Center

for Supercomputing Applications(NCSA) が開発した衛星データを始めとする科学技術系のデータに広く用いられている汎用的なデータフォーマットである．図 2 に示されるように，階層構造を利用して，複数の多様なデータセットを格納することができ，階層構造の各ノードおよびデータセットに対して複数のメタデータが定義できることが大きな特徴である．また，NCSA の提供するライブラリによる HDF ファイルへのアクセスが可能であり，格子データの部分抽出などのライブラリが提供されている．このライブラリはユーザによる拡張も出来る形で配布されている．

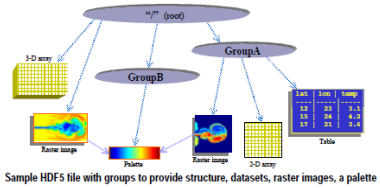


図 2　HDF 構造イメージ

3.2. HDF-EOS

HDF-EOS とは衛星による地球観測データのために NASA と NCSA によって開発された HDF に基づくデータフォーマットである．NASA のデータセンター(DAAC)をはじめとして，多くの衛星データが HDF-EOS にて配布されており，この分野における標準データフォーマットとなりつつある.HDF-EOS には POINTS，SWATHS，GRIDS の3種類のデータ構造が定義されている．Grid データはパラメータと共に1セットの投影方程式(メルカトル図法等)を含んでおり(図3)，Point データは不規則な時間間隔と点在している地理的な位置で観測された一連のデータによって構成されている(図4)．Swath データは,衛星が軌道面に沿って移動しながら軌道面に垂直なスキャンラインを取り，スキャニングをしたものをそのまま保存できるデータ型である(図5)．

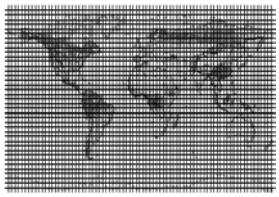
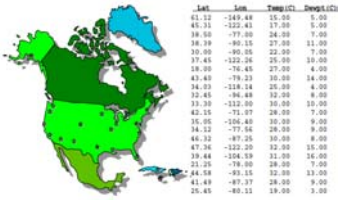


図 3　メルカトル図法



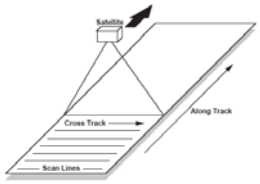


図 4 Point データセット例　図 5 典型的な衛星の Swath

これらのうち Point データと Swath データのデータモデルを次に示す.Grid データについては，Swath データの拡張型であるため割愛する．

【Point データ】

Point データは表形式でデータを格納できる．図6の例では，データの観測地点の情報が入ったテーブルである Desc-loc と観測データの入ったテーブルである Observation があり，Linkage の設定により，フィールド間の関連を定義することができる.2つのテーブルの ID を関連付けることにより，ID001の Chicago の観測点は緯度 41.49，経度-93.37 で,そこで観測されたが Time0800 で-3℃,0900 で-2℃,1000 で-1℃であることがわかる．

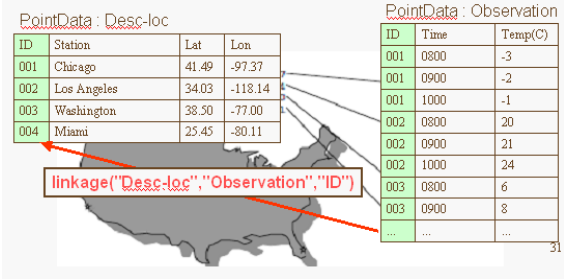


図 6　Point データモデル

【Swath データ】

Swath データは，データ配列の軸となる Dimension が予め数種定義されており，実際に観測値を入れ込むためのデータフィールドを定義する際に Dimension を選択して配列を定義する.Dimension は,名前と大きさをユーザが自由に定義することが出来る.図7では，衛星の進行方向に対して垂直である Xtrack，平行である Ytrack，鉛直である Ztrack の 3 つの Dimension がある．Tempreture というデータフィールドは，Xtrack，Ytrack，Ztrack の 3 つの Dimension を持つ 3 次元格子として定義されている．Time は，進行方向に沿った Ytrack のみの 1 次元格子である．Time，Latitude，Longitude などの位置情報をあらわすデータフィールドを特にジオフィールドと呼ぶ．

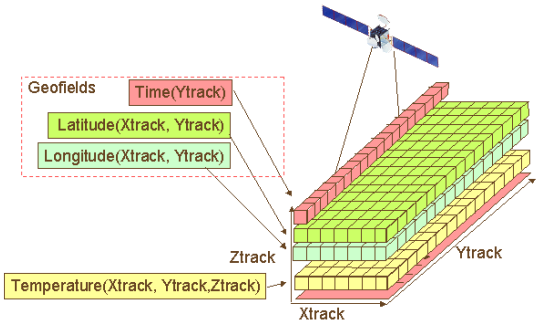


図 7　SWATH データモデル

4. 選択操作関数の実装

本研究にでは，データ構造に基づいた問合せモデルの定義と実装を最終的な目的とする．そのための初期段階として，我々は問合せモデルの基本要素として用いられる選択操作関数を実装した．選択操作は，例えば「180<temperature<220」のように属性値の条件によりデータを絞り込む操作である．

HDF-EOS ライブラリには部分格子の抽出関数は提供されているものの，データ値を条件とした単純な選択操作関数は提供されていない．そのためデータを絞り込む場合は（部分）格子を一度メモリ上に取り出した後，配列の各要素を順次確認しなければならず，巨大なデータから条件にあった部分格子だけを取り出すのに相当な時間と労力が必要である．

巨大な格子データに対してアクセス効率を向上させるためには，あらかじめデータに対して索引を定義しておくのが効果的である．そこで我々は HDF-EOS ファイルに対して高圧縮な Bitmap 索引を埋め込み，単純な条件検索を高速に行うための関数を提供する．

4.1. 関数の利用例 -Swath データを対象に-

HDF-EOS において，選択操作関数を用いた部分データ抽出例として最もよく利用されると考えられるものは，Swath データにおけるジオフィールド値を条件にした部分データの絞込みである．Swath データは衛星軌道に沿った 1 次元から 3 次元の帯状のデータであり，衛星軌道に沿った方向に非常に長い格子を持つ．科学者は帯状の Swath データを緯度軸・経度軸による格子に写像，欠落分を保管してデータの分析を行う．3.2 節で述べたように，ジオフィールドは緯度，経度，時刻などの位置および時間に関する値である．Swath データが衛星の軌道に対して並行，垂直，鉛直の 3 方向を軸とするという特徴から，緯度，経度，時間といったジオフィールドは次元軸として定義されるのではなくフィールド値として定義されている．そのため，例えば「緯度が 25 度以上 45 度以下かつ経度が 125 度以上 145 度以下」という条件による部分格子抽出は，フィールド値による選択関数によって実現される．実際 HDF-EOS ライブラリには，Swath データから緯度経度の範囲を指定して該当する部分データをデータフィールドから取得する関数が用意されているが，その関数ではデータセットをファイルから取得した上で全ての値を検証しており，大規模なデータには適さないと予想される．これについては我々の提供する関数との有用性の比較検証を行う予定である．

4.2. ライブラリの概要

図8にライブラリの構成を示す．提供する関数は，索引生成（および削除）関数（図中①②），選択条件設定関数（図中③），選択操作実行関数（図中④）である．なお，生成する索引の種類およびアルゴリズムに関しては 4.3 節にて，実装した関数を利用した実行プログラム例について 4.4 節で詳細に説明する(より具体的な関数の説明は付録 a 参照のこと)．

また，生成された索引は HDF-EOS ライブラリにより自動的に構築される階層構造の外に IndexInfo（図中⑤）を作ることによって，HDF-EOS 利用者から隠蔽されるようにしている．

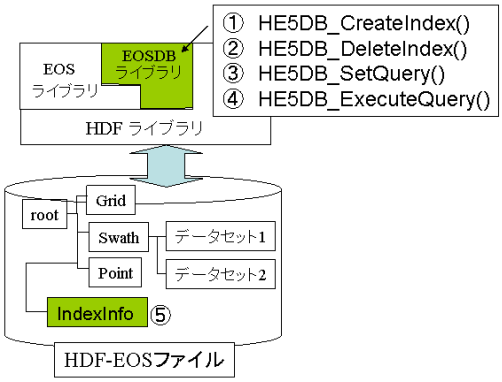


図８：ライブラリの概要

4.3. Bitmap 索引

索引付けの手法として,本研究では Bitmap 索引を用いる．この索引は地球大気科学データのようなデータ更新がなく問合せ結果が大きくなるデータに対して非常に有効である．

Bitmap 索引はある簡単な条件を設定し,それに該当すれば1,でなければ0とする bit 列で表現される.例えば， X を0から9までの整数をもつ配列とする．値の範囲指定のための Bitmap 索引を作成するとき,"X(i)>=p"(p は0～9の整数)に当てはまるものを予め計算する．X が N 個の値を持つとすると，各 Bitmap は N bit の0もしくは1を含む．"X(i)>=1"を表現する Bitmap を作る際，i 番目の値が 1 より大きければ i 番目の bit は1となり，そうでなければ0となる．X の Bitmap を作る際 0～9 に対応する Bitmap が計 10 個作られる．ユーザが"X(i)>=p"という条件で問合せするときは，値を直接検索することなくこれらの Bitmap を読むだけでよい．索引付けをする際の数値の分割数はどのような問合せを行うかによって変わってくるので，ユーザが索引を生成する際に指定できるようにする必要がある．Bitmap 索引は通常 1 つの条件属性に関して定義されるが,複数の条件を含む問合せを処理することも容易である．たとえば，X と Y の 2 つインデックスを持つデータに対して"X(i)>=px and Y(j)>=py"という問い合わせをしたとする．このとき，最初に X と Y のインデックスで部分解を取得し，それらに対して AND 演算を行うことで複数の条件に対する問合せができる．

また，Bitmap 索引を採用した理由として重要な点は,Bitmap 索引を用いた有用な可視化アルゴリズムが確立されていることにある[5]．この論文で用いられている Word-Aligned Hybrid (WAH) code ([6]図 9) による Bitmap 索引の圧縮を，我々が実装する際に用いる．通常 Bitmap の圧縮には Run-length encoding という連続する0または1個数を交互に表記する手法が一般的である．WAH 圧縮では bit 列を 32bit を 1word として分け，16 進数の文字列に変換する．1word が全て同じ bit である場合は Run-length encoding と同様に同じ bit のみの word が連続する数として表記される．0bit または 1bit のまとまりが小さければ，Run-length encoding よりもWAH圧縮の方が充分に

高圧縮となる．この手法は bit 列の空間関係を維持できるため,圧縮したままの索引参照,bit 演算が可能であるため，使用容量，計算量ともに大幅に節約できる．

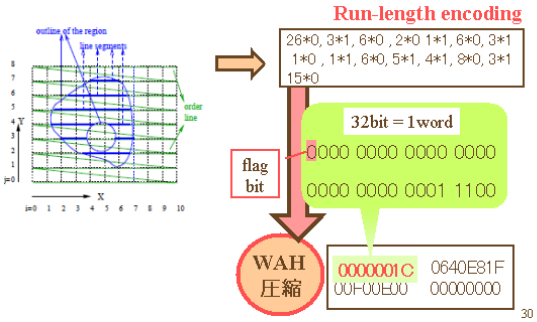


図 9　WAH 圧縮

4.4. 選択関数の利用手順

我々は HDF-EOS のための C 言語による拡張ライブラリを実装し，次の関数を提供する．

1. 処理：索引を生成する
   （しなければ，シーケンシャルスキャンによる検索を行う）
2. 問合せ条件の設定（複数設定することも可能）
3. 問合せ条件の実行

今回の実装では，問合せ条件の記述に関して以下の制限を設け，単純な問合せのみを可能にしている．

- 1 つの Swath データセットに対してのみ検索可能とする
- 問合せの結果として返されるフィールド値はただ 1 つのみとする．

ここで，ファイル”aura.he5”に格納されている”OMI data”という名前の Swath データを用いた問合せ例を示す．このデータの構造は以下の通りである．

経度：Longitude（Xtrack，Ytrack による 2 次元格子）
緯度：Latitude（Xtrack，Ytrack による 2 次元格子）
地上の気温：Temprature（Xtrack，Ytrack，Ztrack による 3 次元格子）

【問合せ例】

緯度が 25 度～45 度，経度が 125～145 度，気温が 200K～250K という条件を満たす格子点と気温データを求める

Step1：前処理（索引を生成する）

まず,問合せ条件で利用する Longitude，Latitude，Temperature の 3 フィールドに対して，索引を生成する．索引を生成する時の各々の条件分割数をそれぞれ 360,180,100 とする．索引生成のための HF5DB_CreateIndex 関数を使用したプログラム例を以下にに示す．

```
#include “HE5_HdfEosDef.h”
#include “he5db.h”
```

```
int main(){
    // HDF-EOS ファイル”aura.he5”を開く
  hid_tfid = HE5_Swopen(“aura.he5”,H5F_ADD_RDWR);
    // SWATH データ”OMI data”をアタッチする
      hid_t swid = HE5_SWattach(fid,“OMI data”);
    //Longitude フィールドに索引を付与する
    HE5DB_CreateIndex(swid,”Temprature”,360);
    //Latitude フィールドに索引を付与する
    HE5DB_CreateIndex(swid,”Temprature”,180);
    //Temprature フィールドに索引を付与する
    HE5DB_CreateIndex(swid,”Temprature”,100);
    //ファイルを閉じる
    HE5_SWdetach(swid); HE5_SWclose(fid);}
```

HF5DB_CreateIndex 関数は3つの引数を持ち，それぞれ索引生成対象となる SWATH データの ID，索引を作成するフィールド名，索引を生成するための条件の分割数となっている．

Step2：問合せ条件を記述する

問合せ条件は，HE5DB_SetQuery 関数にて記述する．また，HE5DB_ConbineQueries 関数にて複数の問合せ条件を論理結合することもできる．問合せ条件を記述するプログラム例を以下に示す．

```
// HDF-EOS ファイル”aura.he5”を開く
    hid_t fid = HE5_SWopen(“aura.he5”,
                                    H5F_ADD_RDWR);
    // SWATH データ”OMI data”をアタッチする
    hid_t swid = HE5_SWattach(fid, “OMI data”);

    //問合せ条件を設定する
     hid_t qid1 = HE5DB_SetQuery(swid,
                                    “Longitude”,”>=”,125);
     hid_t qid2 = HE5DB_SetQuery(swid,
                                    “Longitude”,”<=”,145);
     hid_t qid3 = HE5DB_SetQuery(swid,
                                    “Latitude”,”>=”, 25);
     hid_t qid4 = HE5DB_SetQuery(swid,
                                    “Latitude”,”<=”, 45);
     hid_t qid5 = HE5DB_SetQuery(swid,
                                    “Temperature”,”>=”, 200);
     hid_t qid6 = HE5DB_SetQuery(swid,
                                    “Temperature”,”<=”, 250);

     //問合せ条件を論理結合する
     hid_t q12 = HE3DB_CombineQuery(qid1, qid2,
                                    HE5DB_AND);
     hid_t q34 = HE3DB_CombineQuery(qid3, qid4,
                                    HE5DB_AND);
     hid_t q56 = HE5DB_CombineQuery(qid5,
                                    qid6,HE5DB_AND);
     hid_t qall = HE5DB_CombineQuery(q12, q34,
                                    HE5DB_AND);
     hid_t qall = HE5DB_CombineQuery(qall, q56,
                                    HE5DB_AND);
```

この例では，HE5DB_SetQuery 関数によって設定された問合せ条件を AND 演算によって論理結合している．HE3DB_CombineQuery 関数により 2 つずつ条件を結合していく．

Step3:問合せを実行する

HE5DB_ ExecuteQuery 関数の実行により，条件に該当する格子点の位置とフィールド値が得られる．Step2 にて記述された条件を満たす Temperature フィールドを取得する部分は，以下のように記述する．

```
int **pos, *result;
 HE5DB_ExecuteQuery(qall,"Temperature",&pos,
                                    &result);
```

引数の pos は位置情報を取得するための配列で，result はフィールド値を格納するためのものである．

5. 検証

ここで，本研究での実装関数の性能検証を行う．検証にあたって，米国航空宇宙局(NASA)の AURA 衛星に搭載された OMI(オゾン監視装置)から得られた Swath データ[7]を使用する．この Swath データは，Latitude (nTime* nXtrack) , Longitude (nTime* nXtrack), Time (nTime), Wavelength(nWavel), SO2index(nTime* nXtrack)等で構成されている．

まず，前処理である Bitmap 索引生成プログラムの実行時間および WAH 圧縮による Bitmap 索引の圧縮率をそれぞれ図10，表2に示す．

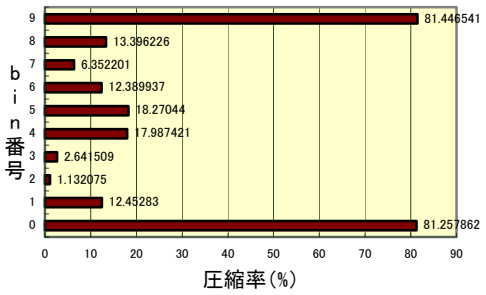


図 10　Bitmap 索引の圧縮率

リアルタイム	0.184s
ユーザ時間	0.040s
システム時間	0.010s

表 2　Bitmap 索引生成の実行時間

これらは，swath ファイルの Longitude フィールド(nTime[1643] * nXtrack[60]の二次元配列)に対して，分割数10の Bitmap 索引を生成した際のものである．緯度を 10 領域に区切った範囲をそれぞれ bin と呼ぶ．各 bin 幅は bin0: -179.998764～-143.998947, bin1: -143.998947 ～ -107.999123,… となる．WAH 圧縮ではひとつの ward が全て0または 1 である場合，word の先頭に flag-bit を立て，同じ状態の連続している数を残りの bit で表し，圧縮する．表1では，圧縮率を【生成された索引の大きさ/その圧縮が一切ない状態の bitmap 索引の大きさ】として表している．

HDF-EOS ライブラリに従来備わっている関数を用いてある緯度範囲内の部分格子を SO2index フィールドから抽出しようとするとき，Latitude フィールドの全格子を検索した後，条件に該当する部分格子のデータセットを取り出すといった手順を踏む．緯度範囲の検索のために使用されるメモリ容量を比較したものを図11に示す．

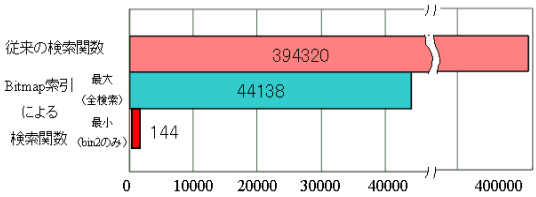


図１１　検索のためのメモリ使用量（B）

従来の検索関数では 394321B (nTime[1643] * nXtrack[60] * float 型[4B])のメモリ領域が，検索の過程で消費されている．対して，bitmap 索引による検索関数では，全検索を行う場合でも 44138B(非圧縮時の最大 word 数[1643*60/31] * 1word の容量[4B] * 圧縮率の合計[247%] + bit 演算結果を入れるための領域[3180*4B])と，従来と比べて飛躍的に小さくなり，検索範囲に該当する bin が一番圧縮されている bin2 であった場合は 144B(index 容量[3180*1.13%]*4B)と非常に縮小される．大規模データの検索の際にはメモリ削減，検索速度の面において，bitmap 索引を用いた領域検索が効果を発揮することが期待できる．

次に，従来 EOS ライブラリに備わっているデータサブセット抽出プログラムの処理速度を計測し，我々の実装した Bitmap 索引を用いて同等の検索を行うプログラムとの速度比較を行った(図12)．

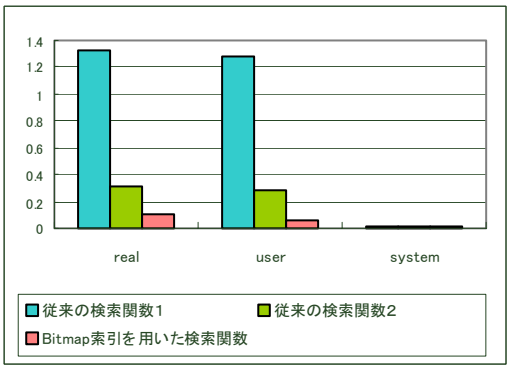


図１２　検索関数の速度比較

これは，「-179.0 <= Longitude <= -143.0, -73.0 <= Latitude <= 89.0 の範囲の領域を "SO2index" フィールドから抽出する」という問合せを行ったものである．問合せ対象のフィールドである Longitude, Latitude, SO2index ともに 1643*60 の2次元配列である．表 5 のうち左側のものが HDF-EOS ライブラリによるサンプルプログラムの検索関数を用いたもので，中央が一度すべての Longitude, Latitude, SO2index データセットを取得してから該当する領域を検索

した場合，右側が今回実装した Bitmap 索引を用いた検索関数の実行速度である．従来の検索関数1と比較して Bitmap 索引を用いた検索関数の実行速度が明らかに小さい．索引の生成時間（表2）を考慮に入れてもなお実行時間的に勝っているという結果が得られた．通常ユーザが検索に使用するものと予測されるサンプルの検索関数は，指定された緯度経度情報をデータフィールドから読み込んで検索領域を取得し，その結果に従って別のデータフィールドを検索するというものである．その実行速度に疑問があり，最低限の作業のみの検索プログラムを別途用意し，速度比較をした．その結果，やはり Bitmap 索引を用いた検索が 3 分の1程度の速度となり，索引付けによる HDF-EOS データ検索の効率化の有用性が示された．

6. まとめと今後の課題

本稿では分析データの大容量化に伴う科学者による DBMS 利用への動きに注目し個人利用における DB 利用を実態を調査した.そしてその考察結果に基いて,HDF-EOS をベースとした簡易データ管理システムを提案し，索引構造を埋め込むための HDF-EOS のための追加ライブラリを実装した．

今後，HDF-EOS ライブラリだけでなく既存の DBMS との性能比較・検証を行い本システムの有用性の基盤を固めていきたい．HDF の仮想ファイルレイヤーという，外部ファイルのデータセットにリンクを張ることができる機能を利用して複数ファイルのデータセットを 1 つのルートファイルでまとめて管理・検索できるようし，外部アプリケーションとの連動を可能にしたい．

文　献

[1] Jim Gray; David T. Liu; Maria A. Nieto-Santisteban; Alexander S. Szalay; Gerd Heber; David DeWitt, “Scientific Data Management in the Coming Decade”, January 2005.

[2] Xiaosong Ma, Marianne Winslett, John Norris, Xiangmin Jiao, Robert Fiedler. "GODIVA: Lightweight Data Management for Scientific Visualization Applications," icde, p. 732, 20th International Conference on Data Engineering (ICDE'04), 2004.

[3] The NCSA HDF Home Page: http://hdf.ncsa.uiuc.edu/

[4] HDF-EOS Project: http://hdf.ncsa.uiuc.edu/hdfeos.html

[5] Kurt Stockiinger, John Shelf, Kesheng Wu, et.al., “Query-Driven Visualization of Large Data Sets”, In Proceedings of IEEE Visualization 2005, pp.167 – 174 (2005)

[6] Kesheng Wu, Ekow J. Otoo, Arie Shoshani, ”Compressing Bitmap Indexes for Faster Search Operations.” SSDBM 2002: pp.99-108(2002)

[7] OMI/Aura Data Project: http://daac.gsfc.nasa.gov/Aura/OMI/

付録 a 実装関数の説明

・HE5DB_CreateIndex(hid_t *swid, char *fieldname, int ndevide);

swid：対象となる Swath データの ID

fieldname：索引を作成するフィールド名

ndevide：索引を生成するための条件の分割数．

・hid_t　HE5DB_SetQuery(hid_t *swid, char *fieldname, char *comp, float value)

hid_t swid：対象となる Swath データの ID

char *fieldname：問い合わせ対象のフィールド名

char *comp：比較演算子(<, >, =, <=, >=, !=, ==)

float value：比較する値

戻り値：問合せ条件に対する ID

（失敗した場合は不の値）

・hid_t　HE5DB_CombinateQueries(hid_t　qid_a,　hid_t qid_b, int OPERATION)

hid_t qid_a：結合する問合せ条件の ID（左辺）

hid_t qid_b：結合する問合せ条件の ID（右辺）

入れない場合は -1 を入れる

int OPERATION：同じ Swath データに対してすでに設定されている

問い合わせ条件との関係

HE5DB_AND：論理積（qid_a AND qid_b）

HE5DB_OR：論理和（qid_a OR qid_b）

HE5DB_XOR：排他的論理和（qid_a OR qid_b）

HE5DB_NOT：論理否定（NOT(qid_a)）

戻り値：結合された問合せ条件に対する ID

（失敗した場合は不の値）

・int HE5DB_ExecuteQuery(hid_t *swid, char *fieldname, int *pos[][], float *result[])

hid_t swid：問合せ対象の SWATH データの ID

char *fieldname：出力対象のフィールド名

int pos[][10]: 格子点の位置（10 次元まで指定可能）

float value[]: フィールド値

戻り値：結果の件数