EC83－49

多次元クラスタリング技法に基づくGRACE二次記憶系の設計と評価

伏見信也・喜連川　優
田中英彦・元岡　達
（東　大）

1984年2月27日

社団法人　電子通信学会

EC83－49

多次元クラスタリング技法に基づく GRACE二次記憶系の設計と評価

Secondary Memory System of Relational Algebra Machine GRACE Based on The Multi-dimensional Clustering Technique : Its Design and Evaluation

伏見 信也† 喜連川 優†† 田中 英彦† 元岡 達†

Shinya Fushimi Masaru Kitsuregawa Hidehiko Tanaka Tohru Moto-oka

†東京大学 工学部 ††東京大学 生産技術研究所

Faculty of Engineering University of Tokyo
Institute of Industrial Science University of Tokyo

1. はじめに

関係データベースマシンGRACE〔1〕は，従来関係データベース処理の隘路となっていたjoin，projection等の処理負荷の重い演算を高速に実行することが出来る。従ってGRACEに於ける問合せ処理の隘路は低速大容量の二次記憶系（磁気ディスクを主体とするディスクモジュール）に対するアクセス，即ちselection 演算処理に移行することが予想され，これを高速化することにより高い性能を実現することが出来る。本稿では問合せ中の検索述語の分布，リレーションのタプルの分布を考慮した多次元クラスタリングアルゴリズムを提案し，二次記憶へのアクセス時間に於いて支配的な，ディスクに対する平均アクセスページ数を大幅に減少させることを試みる。

2. データベースマシンGRACEの構成とその二次記憶系

GRACEは図1に示す様にプロセッシングモジュール（ハードウェアソータを用いた関係代数演算処理系），メモリモジュール（中間記憶系），ディスクモジュール（二次記憶系）及びコントロールモジュール（マシンの統合／制御系）の4種のモジュールから成り，hashとsortを用いた並列アルゴリズムにより，joinすべき2つのリレーションのタプル数を各々T1,T2，当該join演算に割り当てられたメモリモジュールの台数をmとすると，$\mathcal{O}((T1+T2)/m)$ 時間でjoinを実行することが出来る〔1〕。GRACEの二次記憶系は大容量可動ヘッドディスクを有するディスクモジュールであり，ここにベースリレーションが格納される。GRACEではjoin等の演算がタプル数に比例した時間で処理され，ディスクモジュールから送られてくるタプルのストリームに沿った関係代数処理が実現されている。従ってGRACEに於ける関係代数演算の処理性能は基本的にタプルストリームの生成機構としてのディスクモジュールへのアクセス性能，即ち当該演算に必要なタプルを得る為のディスクへのアクセス回数，1アクセスに要する時間，及びディスクからのデータの転送レートで定められる。ディスクからのデータの転送レートはディスクやディスク制御装置自身の物理的特性により規定されるが，一方ディスクへのアクセス回数，

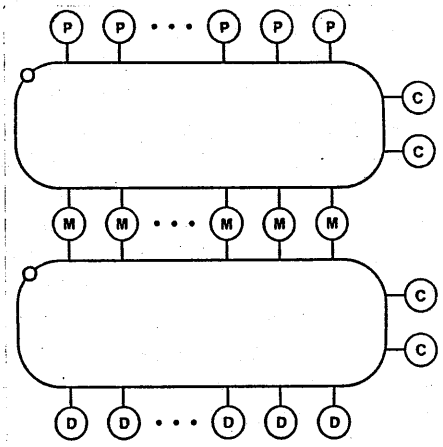


Fig.1 Global Architecture of GRACE

アクセス時間は各々タプルを適切にページ分割すること，更にそれらのページをディスク空間に適切に配置することにより大幅に減少させることが可能である。即ち，一般に二次記憶系の設計問題は，i）リレーションのタプルをディスクのページ容量の大きさに分割するタプルのページ分割問題と，ii）これにより得られたページをディスク内にその機械的特性を考慮しつつ配置するページの物理的配置問題の2つに分類される。ディスクへのアクセス時間を支配するのは検索述語を満たすタプルを得るために必要なディスクへのアクセス回数であり，これは基本的にi）のみによって定まることから，本稿では以下この問題のみについて考える。尚，本稿を通じて問合せとは問合せ中に存在するselection predicate を指すものとする。

3. 多次元クラスタリングと二次記憶系の設計

3.1 多次元クラスタリング

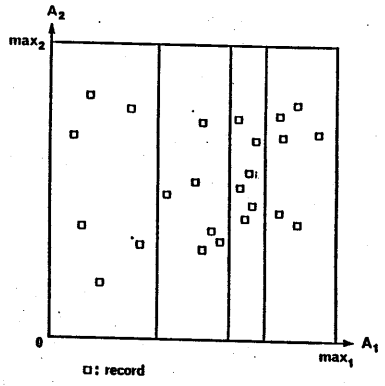
一般にディスクに対するアクセス速度はその物理的特性によ

り半導体記憶のそれに比して極めて低速である。これを補償する為，通常ディスクの記憶空間はページと呼ばれる一定量の単位に分割され，入出力動作はこれを単位として実行される。ここで１ページの容量をＶ〔タプル〕，或る問合せｑによってアクセスされるタプルの個数，及びページ数を各々τ(q)，π(q)とすると，τ(q)はｑとリレーションを構成するタプルによって一意的に定められるのに対し，π(q)はこれに加えてページ空間に於けるタプルの分布に依存する。一般にτ(q)／Ｖ≦π(q)≦τ(q)であるが，τ(q)≒π(q)はｑにヒットするタプルが完全にページ空間に分散してしまっている場合，即ち各ページ中にｑにヒットするタプルが高々１つしかない場合に成立し，一方τ(q)／Ｖ≒π(q)はτ(q)個のタプルがページ空間で必要最低限のページ数τ(q)／Ｖに集積されている場合に対応する。ディスクの入出力動作はページ単位に行われることから，後者の状態はｑに対するディスクアクセス回数が最小の場合であり，ｑに関して理想的なページ分割が行われていることを示し，ページ空間はｑに対しクラスタリング特性を有すると言う。

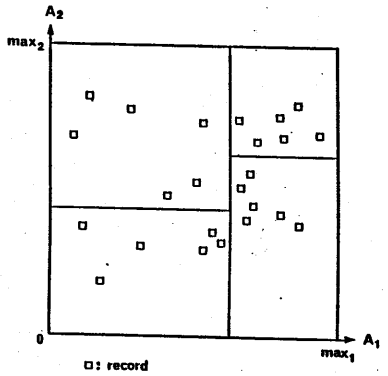
しかし全ての問合せｑに対し完全なクラスタリング特性を実現することは原理的に不可能である。そこでシステム全体の性能を左右する平均アクセスページ数πに着目し，これを最小化することが考えられる。一般にページ化された二次記憶系に於いて，問合せにヒットするタプルを出来る限り少数のページに集積することによって平均アクセスページ数πを減少させる手法をクラスタリングと呼ぶ。従ってクラスタリングの基本戦略は『頻繁にアクセスされるタプルを出来る限り少数のページに格納すること』である。

ここでクラスタリングの観点から従来の二次記憶系設計の設計技法を振り返ってみると，２分木，Ｂ木，ハッシュ等に基づく従来の技法は，基本的に主キーのみに関するページ分割，即ち主キーに関する一次元クラスタリングと見なすことが出来る（図２(a)）。従って，主キーに関する問合せに対しては，単純な値指定のもの（ｖ＝ｃ）から，更に（２分木，Ｂ木等に於いては）range query （c1≦ｖ≦c2）に対しても原理的に最小のページアクセス数を実現することが出来る。一方，非キーに関する問合せに対しては，主キーのみに対するページ分割を行った結果，タプルが非キー値に関してページ空間に分散してしまい，極めて多数のページアクセスが必要とされる。このことは非キーに対するアクセスパスが交代インデックス等の密な木に限られることと対応する。この様に，従来の設計手法は主キーに対するクラスタリング特性を重視しすぎたページ分割を行うことによって，全体として平均アクセスページ数を著しく増大させている。

多次元クラスタリングとは主キーのみでなく非キーを含めた複数の属性に対してページ分割を行うことによって，平均アクセスページ数の減少を図るクラスタリング技法の一つのクラスである（図２(b)）。ここで，(1)多次元クラスタリングは各属性の軸に沿ったページ分割しか許さず，ページはリレーションの原空間（各属性のドメインの直積空間）で必ず超直方体となる点でクラスタリング技法全体の内の比較的小さな部分集合であるが，これよりも大きいクラスタリングのクラスに於いて，無



(a) One Dimensional Clustering



(b) Multidimensional Clustering

Fig.2 One Dimensional Clustering and Multidimensional Clustering

秩序なクラスタリングを行って得られたページ空間に対し効率の良いアクセスパスが存在しない，また(2)多次元クラスタリングは range queryに対し，非常に良好なクラスタリング特性を与える，更に(3)ページ分割を再帰的に行う制約を加えた多次元クラスタリングのクラスに対しては，タプルの挿入，削除等によって生じたページのオーバフロー／アンダフローに対し，ページの分割／併合によって局所的，動的に対処出来る，等の点からこの技法を二次記憶系設計に用いることは妥当であると考えられる。本稿ではこの再帰的な多次元クラスタリングを行うクラスを単に多次元クラスタリングと呼び，このクラスの中での平均アクセスページ数の最小化について検討する。

3.2 コスト評価式

本節では二次記憶系設計の為のコスト評価式として，（多次元）クラスタリングによって生成されたページ空間に対する平均アクセスページ数πを与える式を導出し，更に従来の技法に対するその効果を見る為に，いくつかの例について実際にその評価を試みる。以下，本稿を通じて対象とするリレーションをＲ（A1,...,Ak）（属性総数ｋ），Ｒのタプル数をＴ，ページ数をＮ，ページ容量をＶ〔タプル〕とする。加えて，ここではＲの原空間$D=\prod_{i=1}^{k} \mathrm{dom}(Ai)$〔dom (Ai) は属性Aiのドメイン〕

に於けるタプルの分布関数としてD(t)，問合せ分布関数としてQ(q)を用いる。

1. タプル分布関数　D(t)：

D(t)＝n　⟺　t＝(v1,...,vk) なるタプルがR中にn個存在する。リレーションがトランスポーズされていなければ一般にn＝0 or 1である。即ち，

$$\int_{t\in D} D(t)dt = T$$

であり，D(t)は後に見る様に一定のアルゴリズムが生成可能なページ分割の総数に関与する。

2. 問合せ分布関数　Q(q)：

Q(q)は問合せqに対する正規化された分布関数とする。即ち，

$$Q : \mathbf{Q} \rightarrow [0,1] \quad [\mathbf{Q}はqの全体]$$

$$\int_{q\in \mathbf{Q}} Q(q)dq = 1$$

尚，ここではRのタプルの部分集合としてのページと，Dの空間分割によって得られたDの部分空間の内，このページに属するタプルのみを含むものを同一視し，両者を共にページと呼ぶ。一般にリレーションの原空間DがN個のページ {P1,...,PN} に分割された時，問合せ分布Q(q)に対する平均アクセスページ数$\overline{\pi}$は，各qにヒットするページ数$\pi(q)$にqの重みQ(q)を乗じ，qについてこれを加え合わせることにより得られ，

$$\overline{\pi} = \int_{q\in \mathbf{Q}} \pi(q)Q(q)dq$$

と表される。ここで問合せqによりページPjがアクセスされることをH (Pj,q) と表せば，

$$\pi(q) = \sharp \{Pj \mid H(Pj,q)\}$$

〔$\sharp$は集合のcardinality〕

であり，

$$\delta(Pj,q) = \begin{cases} 1 & H(Pj,q) が真 \\ 0 & H(Pj,q) が偽 \end{cases}$$

と定義すれば，

$$\pi(q) = \sum_{j=1}^{N} \delta(Pj,q) 。$$

従ってこれにより$\overline{\pi}$を次の様に書き換えることが出来る。

$$\overline{\pi} = \int_{q\in \mathbf{Q}} \pi(q)Q(q)dq$$

$$= \int_{q\in \mathbf{Q}} \left(\sum_{j=1}^{N} \delta(Pj,q)\right) Q(q)dq$$

$$= \sum_{j=1}^{N} \int_{q\in \mathbf{Q}} \delta(Pj,q)\, Q(q)dq$$

$$= \sum_{j=1}^{N} \int_{q\in \Gamma(Pj)} Q(q)dq$$

〔$\Gamma(Pj) = \{q\in \mathbf{Q} \mid H(Pj,q)\}$〕

$$= \sum_{j=1}^{N} \mathcal{H}(Pj) \qquad \cdots\cdots\cdots ①$$

但し，$\mathcal{H}(Pj) = \int_{q\in \Gamma(Pj)} Q(q)dq$ 。

即ち，$\overline{\pi}$はN個のページPj各々についてPjにヒットする問合せqの重みQ(q)を加えることによって得られる。この式はクラスタリングのクラスや問合せのクラスに依らず一般的に成立する。ここで，$\overline{\pi}$はQ(q)と {P1,...,PN} のみによって定まることに注意すべきである。

本稿では問合せのクラスとして一般的なrange query の全体 $\{\bigwedge_{i=1}^{k} xi \le vi \le yi\}$ を考察の対象とする。xi≦yiを考慮すれば，range query qは$\prod_{i=1}^{k}$〔xi ,yi〕と同一視され，各qの第i成分は図3に示す直角三角形状の区域中の一点として表現される。ここで各属性のドメインを有界な全順序集合，即ちMINi≦xi∈dom (Ai) ≦MAXi for all iと仮定する。図中，A点は一般のrange predicate に相当し，一方B点，C点は各々predicateが与えられていない場合，値が指定されている場合に対応する。従って，このクラスはrange query のみならず，値指定の問合せ，partial match query をも含む一般的なクラスである。

さて上記の同一視によりqはDの部分集合と見なされるから，

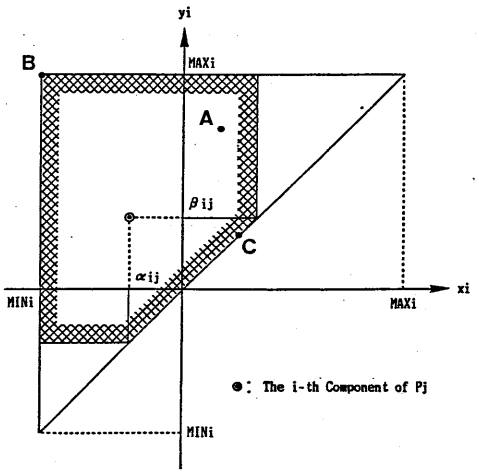


Fig.3 Representation of Range Queries and Page

H (Pj,q) ≡Pj∩q≠φが成立し，更に各ページPjをDに於ける超直方体に限ると（即ち，多次元クラスタリングのクラスに属する技法が生成するページ分割に制限すれば），$Pj=\prod_{i=1}^{k}[\alpha ij, \beta ij]$となり，N個のページは各々Qと同型の空間P内の点として表現される。従って，各PjをQ内の一点として扱うことが出来る（図3）。この時，

$$H(Pj,q) \equiv Pj\cap q\neq\phi \equiv xi\leq \beta ij \wedge yi\geq \alpha ij \quad \text{for all } i$$

から，ページPjにヒットする問合せは各属性について図3中，斜線で示される範囲に属するものとなる。この場合，式①に於ける最左辺は$\overline{\pi}$に対しP内で或るqに着目した式であり，一方式①の最右辺はこれを変形して逆にQ内に於ける各Pjの扱いに着目した式である。

式①の積分値を解析的に求めることは一般に不可能であるが，ここでは特定のページ分割とQ(q)に対する簡単な分布を仮定してこれを実際に計算し，従来の一次元クラスタリング技法に比較して，多次元クラスタリングが実現する性能改善について論じる。

例1. Q(q)は一様な一次元partial match query （k個の属性の内，値が指定される属性がただ一つであり，この属性及びその値がランダムな問合せ分布）とし，各属性に対してDに均等な（$\sqrt[k]{N}$区間に）メッシュ状にページ分割を施して得られたN個のページに対する$\overline{\pi}$を求める。この時，Q(q)は図4に示される様なk個の分布の重ね合わせとして表現される。これから或るページ$Pj=\prod_{i=1}^{k}[\alpha ij, \beta ij]$に対し，

$$\mathcal{H}(Pj) = \sum_{i=1}^{k}\frac{1}{k}\frac{\sqrt{2}\,(\beta ij-\alpha ij)}{\sqrt{2}\,(MAXi-MINi)}$$

$$=\frac{1}{k}\sum_{i=1}^{k}\frac{(\beta ij-\alpha ij)}{(MAXi-MINi)}$$

$$=\frac{1}{k}\sum_{i=1}^{k}\frac{1}{\sqrt[k]{N}}$$

$$=N^{-\frac{1}{k}}.$$

$$\therefore\ \overline{\pi}=\sum_{j=1}^{N}\mathcal{H}(Pj) = NN^{-\frac{1}{k}}=N^{1-\frac{1}{k}}$$

この時，ある属性に対し従来の一次元クラスタリングを施した際の平均アクセスページ数$\overline{\pi}'$は，他に交代インデックス等を持たず，非キーに関する問合せに対し全ページのフルスキャンを行うと仮定しても

$$\overline{\pi}' = (1+(k-1)\cdot N)/k$$

と計算され，$\overline{\pi}$と$\overline{\pi}'$の差はそれ程大きくはない。これは一様な低次元のpartial match query の分布に対し，この様なメッシュ状の多次元クラスタリングを行うと属性が互いに他の属性に対するクラスタリング特性を阻害する状況に陥る為である。多次元クラスタリングは頻繁に参照されるタプルをなるべく少数のページに格納することにより平均アクセスページ数の減少を図る技法であり，本来，Q(q)として多くの属性に対し検索述語が施され，且つその分布に偏りがある場合に最も有効となる。この例に於けるQ(q)はこれら2条件を満たさず，言わば多次元クラスタリングにとって最も不利な分布であり，原理的に大きな性能改善は期待出来ない（4.4 参照）。

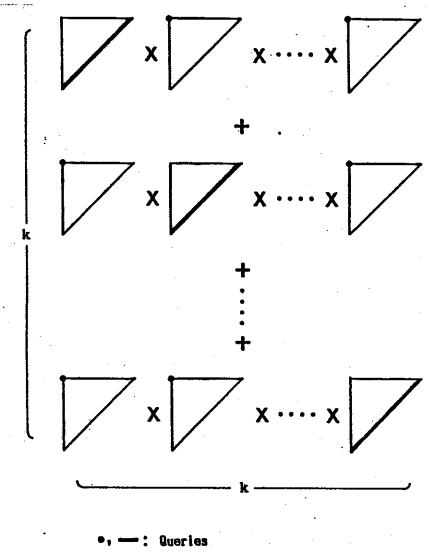


Fig.4 Query Distribution of Uniform One Dimensional Partial Match Queries for K-ary Relaion

例2. Q上Q(q)は一様であり，メッシュ状に分割されたN個のページが与えられた場合を考える。k個の属性が各々ni区間に均等に分割されている（$N=\prod_{i=1}^{k}ni$）とすると，図5に示す様に分割に対応して各属性の射影空間に於ける分割点のシークエンス（$\beta ij' = \delta i\cdot j'$, $\alpha ij' = \delta i\cdot(j'-1)$ $(j'=1,\ldots,ni)$, $\delta i = MAXi/ni$, 簡単の為 $MINi = 0$ for all i とする）が得

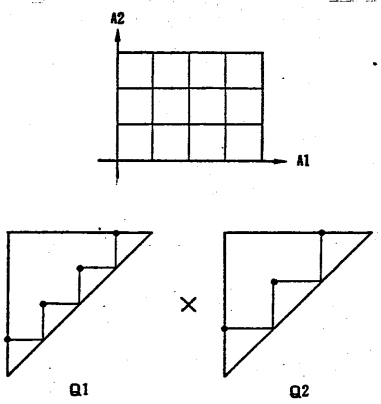


Fig.5 Discrimination Value Sequence for Meshed Space

られる。この場合，各属性に関して図3に於ける斜線部に相当する面積の積がそのまま$\mathcal{H}$(Pj)となる。ページPjの属性Aiに関する図3に於ける斜線部の面積をS(αij, βij)とすれば，

$$\overline{\pi}=\sum_{j=1}^{N}\prod_{i=1}^{k}\frac{S(\alpha ij,\ \beta ij)}{S(0,\ MAXi)}$$

$$=\prod_{i=1}^{k}\sum_{j=1}^{n_i}\frac{S(\alpha ij',\ \beta ij')}{S(0,\ MAXi)}$$

$$=\left(\prod_{i=1}^{k}(2/MAXi^2)\right)*\left(\prod_{j=1}^{n_i}(\delta i\cdot j'\cdot MAXi-\tfrac{1}{2}(\delta i^2\cdot(j'-1)^2+\delta i^2\cdot j'^2))\right)$$

$$=\prod_{i=1}^{k}(ni/3+1-1/3ni)$$

$$\fallingdotseq\prod_{i=1}^{k}(ni/3)=\mathcal{O}(N/3^k)。$$

(if ni >1 for all i)

即ち，この時の平均アクセスページ数$\overline{\pi}$は属性数の3のべき乗に反比例して減少する。これに対し，従来の一次元クラスタリングによる平均アクセスページ数$\overline{\pi}$'は，n1=N,n2=…=nk=1として

$$\overline{\pi}'=N/3+1-1/3N$$

$$=\mathcal{O}(N/3)。$$

これから，この様な場合には多次元クラスタリングによって$\overline{\pi}$は大きく改善されることがわかる。この例ではqの属性，或いはその値に関する分布の偏りは無いものの，一般に全属性に対して検索述語が与えられており，多次元クラスタリング技法はこれを活かして$\overline{\pi}$を飛躍的に減少させている。ここで，あるタプルt=(t1,...,tk)にヒットする問合せqの重みの和ρ(t)は

$$\rho(t)=\lim_{\varepsilon i\to 0}\mathcal{H}([t1,t1+\varepsilon 1]\times\cdots\times[tk,tk+\varepsilon k])$$

$$=\left(\prod_{i=1}^{k}(2/MAXi^2)\right)*\left(\prod_{i=1}^{k}(ti\cdot MAX-ti^2)\right)$$

$$=\left(\prod_{i=1}^{k}(2/MAXi^2)\right)*\left(\prod_{i=1}^{k}(-(ti-MAXi/2)^2+MAXi^2/4)\right)$$

となる。即ち，Q上でqの分布が一様でも，Q(q)をDに重ね合わせた時，直感的にはアクセスパタンは放物曲面と考えてよく，Dの中心(MAX1/2,....,MAXk/2)なる点が最も頻繁にアクセスされる。

3.3 多次元クラスタリングの技法

本節では現在までに提案されている多次元クラスタリングの代表的な技法についてまとめる。このクラスの技法ではページ分割の再帰性からページ空間へのアクセスパスとして木構造のインデックスが自然に導かれる。従って，ここではこの木構造のインデックスとクラスタリングアルゴリズムそのものとを同一視する。

3.3.1 KD-tree

KD-tree (K-Dimensional tree) 法〔2,3〕はリレーションの原空間DとRのタプル分布関数D(t)が与えられた時，先づRのタプルの中からそのA1属性の値が値の大小関係に従ってRのタプル数を2分する様なものを選び，これによりDを2分割する。得られた2つの部分空間D1，Drに対し，各々更にタプル数を2分するタプルを今度は属性A2に関して選択し，これらを各々更に2分する。この操作を分割に用いる属性(分割属性)を巡回的に変化させながら再帰的に繰り返す。ページ容量分のタプルを含む部分空間が得られた時点で各再帰操作は終了する(図6)。アルゴリズムから明らかな様に，本技法はタプル分布D(t)のみに応じたページ分割を行うものであり，Rに対する問合せ分布Q(q)を全く無視している。従って$\overline{\pi}$は多次元にわたるクラスタリングにもかかわらず比較的大きな値となることが予想される。

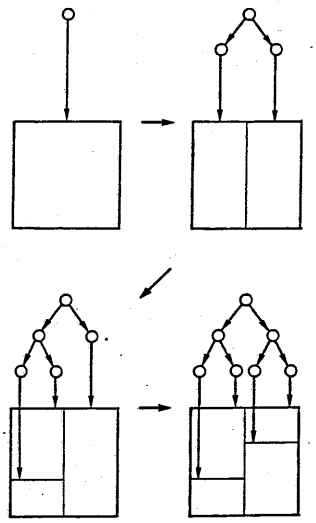


Fig.6 Page Partitioning Process by KD-tree Method

3.3.2 EKD-tree

EKD-tree (Extended KD-tree) 法〔4〕はKD-tree がQ(q)を全く無視したページ分割を行う点を改良し，木の各レベルに於ける分割属性はレベル毎に共通であるものの，レベルに沿った分割属性は固定された巡回的なシークエンスではなく，Q(q)を或る程度考慮したアルゴリズムにより決定する技法である。ここで，対象とする問合せのクラスをpartial match query に限定し，更に問合せ中に値を指定される属性は問合せ分布に於いて確率的に互いに独立であるとすると，$\overline{\pi}$の最小値を与える時の各属性に沿って行われるページ分割の回数が解析的に求められる。Chang らはこの値を基にレベルに沿った分割属性のシークエンスを求めるアルゴリズムを提案した。EKD-tree法は，タ

プル分布に加え，Q(q)を或る程度考慮して元の減少を図るものであるが，アルゴリズムは問合せ分布がpartial match queryの場合に限られており，更にそこでは属性間の確率的独立性が仮定されている。また各レベル内では分割属性は固定されている点などから，range queryを含み，確率的に属性間に強い相関を持つ実際の使用環境下では大きな性能改善は期待出来ないと考えられる。

3.3.3 多次元trie

多次元trieは通常の（binary）trieをKD-tree様に多次元化した技法である〔5，6〕。KD-tree,EKD-treeがページ分割の際にタプル数を2分する値を分割値として選ぶのに対し，多次元trieでは分割値として分割しようとする空間自身を2等分する値を選択する。多次元trieの各ノードは分割属性とその属性に対するビットポジションを持つ。分割に使用するビットは各属性につき最上位から順に用いられる。この技法によるページ分割のアルゴリズムの概略は次の様になる。与えられた空間に対し，一定のアルゴリズムにより分割属性を一つ選びこの属性の未使用ビットの中の最上位ビットの値の0，1に従って空間を2つの部分空間に分離する。生成された2つの部分空間に対して，各々の空間がページ容量個数以下のタプルを含む様になるまでこの操作を再帰的に繰り返す。多次元trie法は，そのアルゴリズムから明らかな様に生成される各ページの境界は2べキ値に限られ，この精度でしかタプル分布に対応出来ない為，一般に100 %のロードファクタは得られない。一方，アクセスパスに必要な補助記憶の容量は非常に小さく保つことが出来る。

分割属性を決定するアルゴリズムとして，KD-treeの様に巡回的にそのシークエンスを定めるものも考えられるが，〔6〕では問合せのクラスを一様なpartial match queryに限定した際の動的ファイル生成（タプルを順に挿入することによりページ分割を動的に行ってファイルを生成する）に対するアルゴリズムを提案している。このアルゴリズムはタプルの挿入によって生じたページオーバフローの際に，分割後のページ空間に対する元が最小となる分割属性を決定するものである。

一般のQ(q)に対して，元の最小化を考えた多次元trieによるページ分割のアルゴリズムはまだ提案されていない。

3.4 クラスタリングのクラス

前節で見た様に，多次元クラスタリングに基づく二次記憶系設計に対してはいくつかの技法が提案されているが，それらの技法はページ分割の際の分割属性及び分割値選択に関する自由度の点から生成可能なページ分割の全体を制約していることがわかる。簡単の為，タプルの分布D(t)はDに対し一様であるとすると，リレーションRをNページに分割する時，KD-treeでは分割属性，分割値の選択に関して自由度が無い（アルゴリズムによってアプリオリに定まっている）為，生成可能なページ分割の総数は1となる。EKD-treeでは木の各レベル毎に分割属性選択の自由度があり，ページ分割全体の総数は$k^{\log_2 N}$（D(t)の一様性から木は平衡する）である。多次元trieは木の各ノードについてk通りの自由度がある為，ページ分割全体の総数はk^{N-1}となる。逆に言えば，各々のページ分割アルゴリズムは，この生成可能なページ分割全体の集合の中から出来るだけ小さな元を与える1つのページ分割を選択するアルゴリズムと考えられる。ページ分割の自由度の点から考えれば，木の各ノードに於いて分割属性，分割値共に最大限の自由度を与え，クラスタリングによって生成可能なページ分割全体の集合を拡大することによって実現可能な元の最小値の下限を大幅に小さくすることが可能である。

4．一般化KD-treeを用いたＧＲＡＣＥの二次記憶系の設計

4.1 一般化KD-tree

一般にｎページのタプルを含む空間の分割に対して100%のロードファクタを保証しようとすると，ｋ個の各属性に対しｎ－1通りの分割値候補点が存在する。ページ分割の際の分割値選択に関し，このｋ･（ｎ－1）の自由度を実現する多次元クラスタリングのクラスをここでは一般化KD-tree（Generalized KD-tree,略してGKD-tree）と呼ぶ。GKD-treeによるページ分割の概略は次の様になる。Ｎページ分のタプルを含むリレーションＲのページ分割に対し，一定のアルゴリズムに従ってｋ･（ｎ－1）個の分割値候補点の中から1つを選ぶ。これによって決定されるページ容量ｎに従って原空間Ｄを各々ｎページ，Ｎ－ｎページ分のタプルを有する2つの部分空間Ｄl，Ｄr に分割する。次に得られた部分空間Ｄl，Ｄr に対し，Ｎを各々ｎ，Ｎ－ｎとして手続きを再帰的に繰り返す。分割すべき空間に含まれるタプル数が1ページ容量に達した時点で各再帰手続きは終了する。GKD-treeによって生成可能なページ分割の全体集合の大きさＣ（Ｎ）は漸化式

$$C(N)=\sum_{i=1}^{N-1} k\,C(i)\cdot C(N-i)$$

によって定められ，その値は

$$C(N)=\begin{cases}0 & (N=0)\\ (1/N)\cdot(2N-2)\ C(N-1)\cdot k & (N\geq 1)\end{cases}$$

$$\fallingdotseq (4k)^{N-1}/N\sqrt{\pi\,(N-1)}$$

と求められる。この値は上述のいづれの技法よりも大きく，問合せ分布Q(q)を考慮し，ページ分割の際の分割属性，分割値を決定するアルゴリズムを適切に設定してこの自由度を活かすことにより，他の技法に比較して非常に小さな元を実現するページ分割を得ることが可能となる。

我々はＧＲＡＣＥの二次記憶系設計に対し，そのクラスタリングのクラスとしてGKD-treeを採用することとした。その理由は以下の通りである。

(1) GKD-treeは3.1 (2)(3)で述べた多次元クラスタリング技法が有する利点を保存する。

(2) GKD-treeはページ分割に対し分割属性，分割値選択に関する自由度を可能な限り許すことによって，平均アクセスページ数を大幅に小さくすることが出来る。

(3) ＧＲＡＣＥは大規模関係データベース処理専用マシンである。従ってアクセスパスとして生成されたGKD-treeを専用に設けた半導体記憶中に保持することとしても，コスト的にそれ程問題とならない。更にこれから木の平衡化はさほど問題とならず，平衡化のひきおこすページ分割への制約は緩和されることになる。

4.2 ページ分割アルゴリズム

GKD-treeを用いたクラスタリングを行うにはページの再帰的な分割に際して分割属性，分割値を決定するアルゴリズムが必要である。

さて，前述の様に二次記憶系設計に於けるクラスタリング技法の基本的戦略は『頻繁にアクセスされるタプルは出来る限り同一ページに収める』ことである。即ち，この様なタプルを同一ページに収めることが出来れば，アクセス頻度の少ないタプルがページ空間に分散し，多くのページアクセスを必要としても全体として平均アクセスページ数$\overline{\pi}$は大きく減少する。これを逆に考えれば問合せ分布のpeakにヒットするタプルは出来る限り複数ページに分離しないページ分割，即ちQ(q)の値が小さいqにヒットするタプルから順にページに分割して行くアルゴリズムが考えられる。

ここで，ページ$Pj=\prod_{i=1}^{k}[\alpha ij,\ \beta ij]$のAi属性に関し値$\tau ij$（$\alpha ij\leq\tau ij\leq\beta ij$）でページ分割を行うと，$Pj1(\tau ij)=[\alpha 1j,\ \beta 1j]\times\cdots\times[\alpha ij,\ \tau ij]\times\cdots\times[\alpha kj,\ \beta kj]$及び$Pjr(\tau ij)=[\alpha 1j,\ \beta 1j]\times\cdots\times[\tau ij,\ \beta ij]\times\cdots\times[\alpha kj,\ \beta kj]$なる2つのページが生成される。この時次の定理が成立する。

〔定理〕

$$\mathcal{H}(Pj)+\mathcal{H}(Pj1(\tau ij)\cap Pjr(\tau ij))=\mathcal{H}(Pj1(\tau ij))+\mathcal{H}(Pjr(\tau ij)) \quad \cdots\cdots\cdots②$$

或いは，$\mathcal{H}(Pj)$を$\mathcal{H}(\alpha 1j,\ \beta 1j,\ldots,\alpha kj,\ \beta kj)$と書くと，

$$\begin{aligned}&\mathcal{H}(\alpha 1j,\ \beta 1j,\ldots,\alpha ij,\ \beta ij,\ldots,\alpha kj,\ \beta kj)\\&+\mathcal{H}(\alpha 1j,\ \beta 1j,\ldots,\tau ij,\ \tau ij,\ldots,\alpha kj,\ \beta kj)\\&=\mathcal{H}(\alpha 1j,\ \beta 1j,\ldots,\alpha ij,\ \tau ij,\ldots,\alpha kj,\ \beta kj)\\&+\mathcal{H}(\alpha 1j,\ \beta 1j,\ldots,\tau ij,\ \beta ij,\ldots,\alpha kj,\ \beta kj)\\&\qquad(\alpha ij\leq\tau ij\leq\beta ij)\end{aligned} \quad \cdots\cdots\cdots③$$

〔証明〕属性Aiに関するQ(q)の射影空間Qi(q)を考える（図7）。τijで分割を行うと，図に示される様にAiに関しては$[\alpha ij,\ \beta ij]$が2つのページ境界$[\alpha ij,\ \tau ij]$及び$[\tau ij,\ \beta ij]$に分離される。この分割によって生成された2つのページに対し，Ai以外の属性に関しての計算の積分区域に変更は無く，Aiのみに対して各々積分区域が図中▒，▓で示される範囲に変更される。2つの積分区域は図中■で示される部分が重複しており，この部分は仮想的にページ$PjΔ(\tau ij)=[\alpha 1j,\ \beta 1j]\times\cdots\times[\tau ij,\ \tau ij]\times\cdots\times[\alpha kj,\ \beta kj]$に対応する。従って，積分の積分区域に関する加法性により式②，③が成立する。□

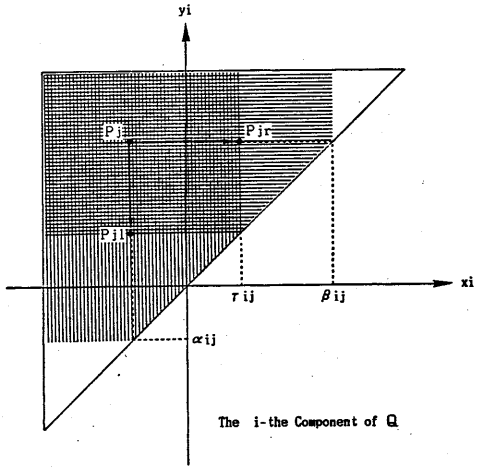


Fig.7 Relationship between Page Splitting and Page Boundaries

上式からPjのページ分割前の平均アクセスページ数

$$\overline{\pi}=\sum_{j=1}^{N}\mathcal{H}(Pj)$$

はPjのτijに関するページ分割によって

$$\begin{aligned}\overline{\pi}'&=\mathcal{H}(P1)+..+\mathcal{H}(Pj-1)+\mathcal{H}(Pj1(\tau ij))\\&\quad+\mathcal{H}(Pjr(\tau ij))+\mathcal{H}(Pj+1)+..+\mathcal{H}(PN)\\&=\overline{\pi}+\mathcal{H}(PjΔ(\tau ij))\end{aligned}$$

に変化することがわかる。即ち，このページ分割によって$\overline{\pi}$は$\mathcal{H}(PjΔ(\tau ij))$だけ増加する。頻繁にアクセスされるタプルを横切ってPjを分割するとこの値は大きくなることから，先の抽象的アルゴリズムは次の様に具体化される。Nページ分のタプルを含むリレーションRの分割に対し，k*(N－1)個の分割値候補点の内，値$\mathcal{H}(PΔ(\tau i))$が最小となるτiを選ぶ。τiによってDが各々N－n，n個のタプルを含む2つの部分空間$D1=[MIN1,MAX1]\times\cdots\times[MINi,\tau i]\times\cdots\times[MINk,MAXk]$及び$Dr=[MIN1,MAX1]\times\cdots\times[\tau i,MAXi]\times\cdots\times[MINk,MAXk]$に分けられたとすると，DをD1（Dr），Nをn（N－n）としてこの操作を繰り返す。各再帰ステップは分割すべき空間がページ容量分のタプルを含む場合に終了する。この手続きをPASCAL風のプログラム言語で記述したものを図8に示す。

本アルゴリズムはバックトラックを行わない直線的アルゴリズムであり，個々のページ分割のステップで局所的に最適な分割値を選ぶものであるから，生成されたページ分割に対する$\overline{\pi}$は必ずしも最小とは限らない。しかし，本アルゴリズムはGKD-treeのページ分割に際しての分割属性，分割値の選択に関して

```
procedure partition(page,page_num);

/* "page" is k-dimensional hyperrectangle space and divided      */
/* into "page_num" pages.                                        */
/* Variable "page" is of the form                                */
/* (low_bound_1,high_bound_1)x.....x(low_bound_k,high_bound_k)   */
/* and                                                           */
/* page[i] = (low_bound_i,high_bound_i).                         */

begin

  Used variables:
    i,partition_attribute       : attribute identifier;
    j,partition_page_num        : page count;
    min_H_value,H_value         : H_value;
    low,high,pv,partition_value : page bound;

  if (page_num == 1) then begin
    Allocate leaf node, and process it appropriately;
    return;
  end;

  min_H_value = sufficient large value;

  /* Find partition attribute and partition value that give      */
  /* minimum H_value.                                            */

  for each attribute i do begin
    (low,high) = page[i];
    for j = 1 to page_num-1 do begin
        pv = Find_Partition_Value_Candidate(page,page_num,i,j);
        page[i] = (pv,pv);
        H_value = Compute_H_Value(page);
        page[i] = (low,high);

        if(H_value<min_H_value) then begin
                min_H_value    = H_value;
                partition_attribute = i;
                partition_value     = pv;
                partition_page_num  = j;
        end;
    end;
  end;

  Allocate internal node and arrange links;
  Partition attribute of this node = partition_attribute;
  Partition value of this node  = partition_value;

  (low,high) = page[partition_attribute];
  page[partition_attribute] = (low,partition_value);
  partition(page,partition_page_num);

  page[partition_attribute] = (partition_value,high);
  partition(page,page_num-partition_page_num);

  page[partition_attribute] = (low,high);
end;
```

Fig.8 Page Partitioning Algorithm for GKD-tree

の大きな自由度を活かし，『頻繁にアクセスされるタプルは可能な限り複数ページに分割しない』という点を保証しており，平均アクセスページ数を大きく減少させることが期待出来る。尚，$\mathcal{H}(Pj\Delta(\tau))$ の最小値を与えるτが複数個存在する場合には，次の様な分割点の決定手続きを採る。

(1) この様なτが或るAiについて複数個存在する場合には，それらの内タプル数を2分する値に最も近いものを選ぶ。これは$\overline{\tau}$の増加に関して変化が無い場合，木をなるべく平衡させ，探索時間を短縮する為である。

(2) この様なτが複数属性について各々複数個存在する場合には，属性間で分割の対象性を与える為親ノードの分割属性の次の属性を選び，これに対して(1)を適用する。

4.3 ページ分割アルゴリズムの時間計算量

$\mathcal{H}(Pj\Delta(\tau))$ の値を計算する手間を1とすると，Nページのページ分割に対し前節で提案したアルゴリズムの時間計算量TC（N）は次の様に求められる。

〔最良の時〕　ページ分割により空間が常に2等分される場合に時間計算量は最小となり，

$$TC(N) = k\cdot(N-1) + 2TC(N/2)$$

即ち，

$$TC(N) = k\cdot((\log N-1)\cdot N+1)$$
$$= \mathcal{O}(kN\log N)。$$

この時対応して生成されるGKD-treeは完全に平衡する。

〔最悪の時〕　ページ分割により生成される空間の一方が常に1ページ容量のタプルを含む場合，最も大きい時間が必要とされ，この時，

$$TC(N) = k\cdot(N-1) + TC(N-1)$$

即ち，

$$TC(N) = kN\cdot(N-1)/2$$
$$= \mathcal{O}(kN^2)。$$

対応して生成されるGKD-treeは直線状のものとなり，木の探索の時間も増大する。

4.4 ページ分割アルゴリズムの評価

本節では4.2 節で提案したページ分割アルゴリズムをいくつかのQ(q)，D(t)に対し実際に実行し，得られたページ分割に対する平均アクセスページ数$\overline{\tau}$をKD-tree のそれと比較することによってその評価を行う。

Ⅰ　Q(q)とD(t)の値の分布に対し，そのpeakにずれがある場合

図9(a)(b)（k＝2の場合）に示す様に，Q(q)のpeakをDの中心に，またD(t)のpeakをDの周辺に生成し，アルゴリズムを実行した。図に示される様にKD-tree ではタプル数を常に2分するページ分割を行う為，Q(q)の分布のpeakを横切ってしまうページ分割が生成される。一方，GKD-treeはQ(q)のpeakを認識し，それを避けつつ，これを囲む様にしてページ分割を行っている。Q(q)のpeakはほぼ2ページに収められている。k＝4の時の平均アクセスページ数$\overline{\tau}$とページ数Nとの関係を図9(c)に示す。これからGKD-treeによるページ分割はKD-tree に比して常にほぼ半分の平均アクセスページ数を与えており，大きな性能改善が得られていることがわかる。更に，Nが大きくなるにつれて$\overline{\tau}$はNの数%程度となり，その絶対値も極めて小さい。

Ⅱ　Q(q)の分布に属性間で偏りがある場合

D(t)はⅠと同様とし，一方Q(q)はk個の属性の内，半数をⅠと同様とし，残り半数は非常に狭い，一様なrange query とする。k＝2の場合のページ分割の結果を図10(a)(b)に示す。この場合，縦軸で示される属性に対して狭いrange query が頻繁に発行されており，この属性に関してページ分割を行ってもほとんど$\overline{\tau}$は増加しない。GKD-treeではこの性質が活かされ，ほとんどのページ分割が横方向に行われているのに対し，KD-tree ではこれを利用することが出来ない。k＝4の時の平均アクセスページ数$\overline{\tau}$とNとの関係を図10(c)に示す。GKD-treeではほと

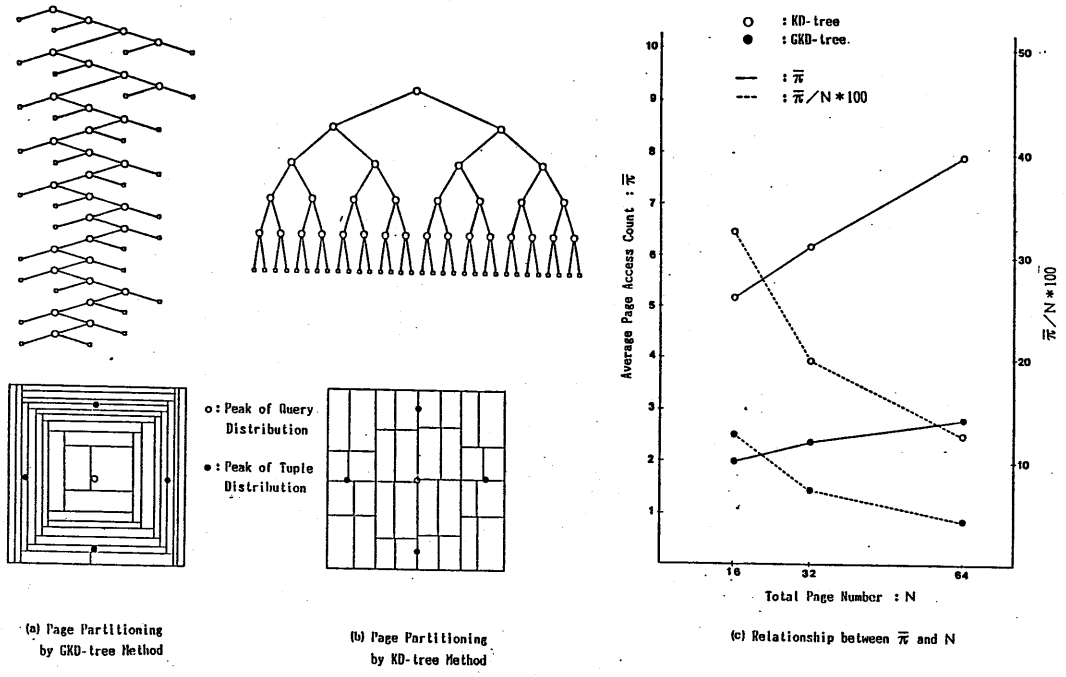


Fig.9 Performance Comparison between KD-tree and GKD-tree (I)

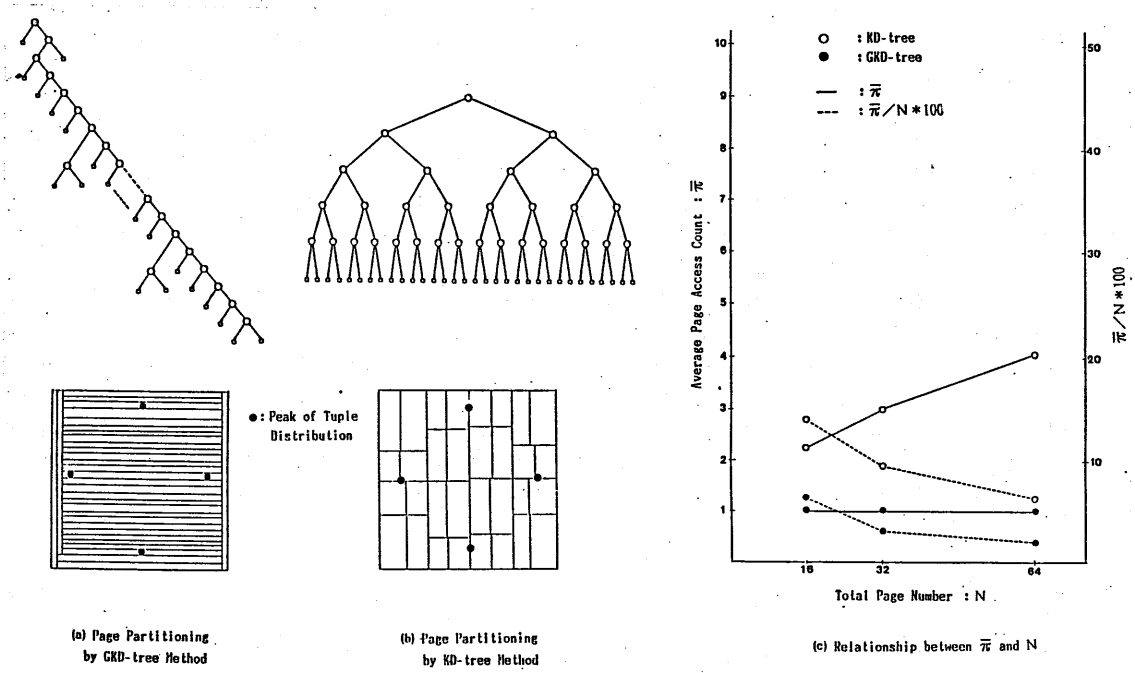


Fig.10 Performance Comparison between KD-tree and GKD-tree (II)

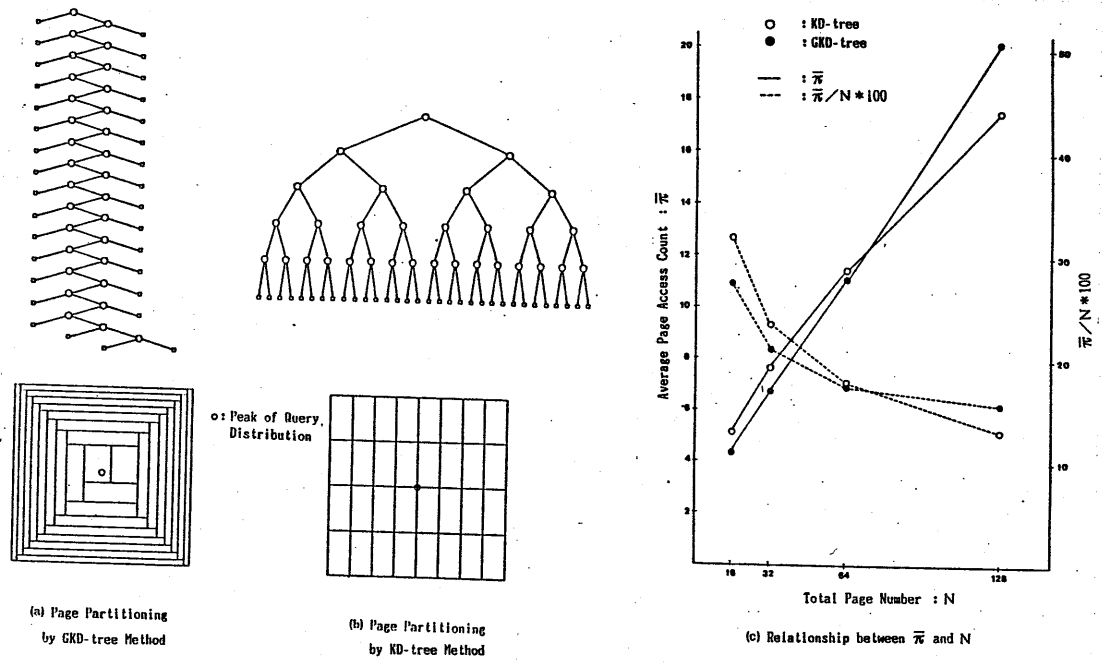


Fig.11 Performance Comparison between KD-tree and GKD-tree (Ⅲ)

んど常に$\bar{\pi} \fallingdotseq 1$であるのに対し，KD-tree ではNに対し線型的に$\bar{\pi}$が増加する。従って，この場合にもGKD-treeはKD-tree に比較して非常に良好な性能を実現している。

Ⅲ　D(t)，Q(q)が共に一様な場合

この様なD(t)，Q(q)に対しk＝2の時のページ分割の結果を図11(a)(b)に示す。3.2 で注意した様に，Qで一様な問合せ分布Q(q)はこれをDに重ねて見た時には，そのアクセスパタンは原点にpeakを持ち，従ってGKD-treeではこれを避けながらページ分割を行う。k＝4の時の$\bar{\pi}$とNの関係を図11(c)に示す。これからNが比較的小さい時はGKD-treeがKD-tree に比べて小さい$\bar{\pi}$を与えるが，Nが大きくなるにつれてKD-tree が有利となることがわかる。これは次の様に説明される。Q(q)は原点にpeakを持つがその形状は3.2 例2で見たように放物曲面である。従ってpeak付近はほぼ平坦であり，逆にドメインの最小，最大値付近では勾配は急になっている。従ってpeakを避けながらページ分割を行っても，初めの数ページの分割によって生成されたページに対する$\mathcal{L}$の値は非常に小さいものの，その後生成されるページに対する$\mathcal{L}$の値はQ(q)がほとんど平坦である為急速に増大し，この効果を相殺してしまう。

Ⅳ　D(t)は一様，Q(q)が一様な一次元partial match query の場合

この場合は4.2 の最後で述べたアルゴリズムのステップによってGKD-treeはKD-tree と同一のページ分割を生成する。3.2 で触れた様に，この場合，多次元クラスタリングによる大きな性能改善は原理的に不可能である。

以上の結果から次の様な結論が得られる。

(1)　D(t)とQ(q)のpeakにずれがある場合，GKD-treeのページ分割アルゴリズムはこの性質を充分利用し，KD-tree によるページ分割に比較して$\bar{\pi}$を大幅に小さく出来る。

(2)　Q(q)の分布に於いて属性間の偏りがある場合，GKD-treeはこの性質を充分に活用し，KD-tree に比較して非常に小さい$\bar{\pi}$を実現することが出来る。

(3)　Q(q)が一様の場合，即ちQ(q)がDの周辺から鋭く立ち上がってDの中心部ではほとんど平坦である場合，ページ数が比較的小さければGKD-treeが有利であるが，ページ数が大きくなるとこれが逆転する。この事実は，Q(q)の分布がD(t)上平坦である場合にはKD-tree が生成する様なメッシュ状のページ分割が有利であることを示している。一方，分割値選択の際に$\mathcal{L}$の値の比較の精度を落とし，4.2 の(1)(2)を適用すれば，これを先のアルゴリズムに埋め込むことが出来る。これによって，ページ分割は先づQ(q)のpeakを囲みながらおこなわれ，peakの頂点付近に達してほぼQ(q)がD上平坦となると，その後はKD-tree 様のページ分割を行って行くことになり，$\bar{\pi}$は更に改善されると考えられる。

(4)　一様な低次元のpartial match query に近い分布に対しては，活かすべき問合せのconjunctiveness が無く，更に個々の問合せにヒットするタプルが相対的に少ない為，タプル間の類似度，近さを判定出来ない。実際，この様な分布に対しては先の$\mathcal{L}$ (PjΔ (r)) の値に差は無く，実質的に4.2 の(1)，(2)のみによってページ分割が行われる。即ち，この様な場合タプル方向にクラスタリングする限りはどの様にクラスタリングを施しても$\bar{\pi}$を大きく減少させることは不可能である。　これを改

善する為には多次元クラスタリングの様なリレーションの横方向の分割だけでなく，リレーションの縦方向の分割（トランスポーズ，即ちDの射影分解）と本技法を組み合せることが有効となる〔7〕。

(5) 以上の(1)～(3)の例では，Q(q)の分布のpeakを避けつつ，これを囲みながらページ分割を行う為，生成されるGKD-treeはほぼ一次元状のリスト構造となり，また頻繁にアクセスされるページは常に木の最深部に位置する。アクセスパスとしてのインデックスがこの様に平衡しないことは通常避けるべきことであるが，ここではGKD-treeをディスクに付加された半導体記憶中に格納することを仮定しており，また頻繁にアクセスされるページは必ず木の最深部に存在することを逆に利用して探索を高速化することも可能である。従ってこの様な場合に於いても，基本的にGKD-tree探索に要する時間は充分に小さいものと考えることが出来る。但し，問合せの分布のpeakが複数個ある場合にはGKD-treeのページ分割アルゴリズムはこれを避けながらページ分割を行う為，生成されるGKD-treeはほぼ平衡木に近い形となる。

4.5 ページ分割アルゴリズムの実現手法

図8に示したページ分割のアルゴリズムの実現に於いては，2つの関数Find-Partition-Value-Candidate，及びCompute-H-Value を効率良く実行することが必要である。前者は分割値候補点を生成し，後者は与えられたDの部分空間に対するℋの値を計算する。即ち，前者はタプル分布関数D(t)に，後者は問合せ分布関数Q(q)に各々直接関与し，これら2つの関数はページ分割アルゴリズムに必要な2つの分布情報の操作手続きとなっている。

4.5.1 分割値候補点の生成

関数Find-Partition-Value-Candidateは，呼び出される毎に分割値候補点を1つずつ生成する。即ち，この処理は次の様に具体化される。

Find-Partition-Value-Candidate (P,n,i,m)

n個のタプルを含むページ$P=\prod_{i=1}^{k}[\alpha i, \beta i]$に対し，属性Aiの値の順序に関してP内のタプルをソートし，mV個めのタプルのAiの値を求める。

ページ分割の初期の段階に於いては基本的にはリレーションのほぼ全タプルのソートが必要とされる。しかし，候補点は一つの属性に対して昇順に要求され，また所要の分割値候補点を求めるにはその近傍のタプルのみを見ればよいことから，ディスクに一定容量のバッファを設け，必要なタプルのみをディスクから読みだし記憶することによりこのアクセスローカリティを活かした効率の良い処理を実現することが出来る。更にソート処理自身に対しては，我々が既に開発したハードウェアソータ〔8〕を用いてこれを高速に実行することが出来る。

4.5.2 問合せ分布情報保持の為のデータ構造

関数Compute-H-Value,即ちℋの計算は，実際には問合せ分布Q(q)を問合せ$q=\prod_{i=1}^{k}[xi, yi]$とその出現回数をエントリとする表として表現し，当該ページ自身を検索predicate として問合せをこの表に対して発行して，ヒットしたエントリの出現回数フィールドを加え合わせることによって実現される。一方で実際のデータベース使用環境では，問合せの到着頻度が1000／sec 程度であることが予想され，全ての問合せをそのまま表のエントリとして保持することは不可能である。従って『一定量の記憶により出来る限り正確なQ(q)の分布情報を得る』ことが必要となる。一般に平均アクセスページ数元の主要項に相当するページはQ(q)のpeakに近いページであるから，頻繁に発行される問合せに対しては正確な情報を必要とし，逆にQ(q)が小さい部分に相当するタプルはどの様にこれをページ分割しても元をそれ程増加させることは無く，この様な部分に対してはそれ程正確なQ(q)の分布情報が必要とされない。そこでQ(q)を保持するデータ構造としてq＝Π〔xi，yi〕に対しxl,yl,…,xk,ykおよびqが発行されたカウント数を1エントリとする表を考え，表のエントリ数が一定量を越えた場合には，カウントの少ないタプル同志をその下位ビットを順にdon't careビットとすることによって併合し，新たに空き領域を生成する技法を提案する。ここではこの様にして得られた分布情報を縮退統計（Gradated Statistics ）と呼ぶ。縮退統計のアルゴリズムを。簡単な具体例で示す。今，次の様な（値，カウント）のペアがあるものとする。但し値は2進のビットパタンで表す。

0 0 1 0 1 1 1	1
0 0 1 0 1 1 0	1
0 0 1 0 1 0 0	2
0 0 1 0 0 0 0	1 9

表のエントリ容量は4であるとすると，これ以上のエントリの挿入はオーバフローとなる。ここで例えば

0 0 1 0 1 0 1　　1

なるエントリを新たに挿入したい場合には，カウントが少ないエントリを選び最下位ビットをdon't careとすることによって2つのエントリを縮退，併合する。この例では次の様になる。

0 0 1 0 1 1 X	2
0 0 1 0 1 0 0	2
0 0 1 0 0 0 0	1 9
0 0 1 0 1 0 1	1

但し，Xはdon't careビットを表す。多次元化された表に対する縮退化の例を図12に示す。図から明らかな様に，この操作によってカウント数の大きい，即ち分布のpeak付近の情報はほとんど縮退することはなく，上記Q(q)に対する精度への要求が満たされていることがわかる。例えば前節の例Ⅰに於けるQ(q)に縮退操作を施し，ページ分割を行った際のr の値の変化は極めて小さいことが確認された。

尚，縮退操作に要する時間計算量は表のエントリ容量をn，エントリの次元をkとすると，$\mathcal{O}(kn^3)$となり比較的大きな手間を必要とする。そこで実現にあたっては縮退統計表の全

500 Data Items
Buffer Capacity = 16 Data Items

Gradated Statistics (val : gradated value, gb : gradated bit)

val gb	val gb	val gb	val gb	count
1fff[13]	3ff[10]	7f[0]	3ff[10]	29
1ff[9]	3f[5]	1f[4]	7f[7]	23
1ff[9]	1f[5]	3f[5]	7f[7]	31
3ff[10]	ff[8]	17[2]	7f[7]	44
3ff[10]	1fff[13]	7f[7]	3f[0]	27
1f[4]	ff[8]	1f[5]	1f[5]	45
7f[6]	f[4]	7f[7]	1f[5]	7
3fff[14]	fff[11]	7f[5]	fff[12]	4
3f[5]	7f[7]	f[4]	7[3]	22
ffff[16]	1fffff[17]	ff[8]	1ff[8]	15
f[4]	1[1]	1f[5]	7[2]	58
3f[6]	3f[6]	f[3]	1f[5]	59
1ff[9]	ff[8]	1f[5]	f[3]	26
7ff[10]	1ff[9]	7f[7]	ff[8]	25
f[4]	1[1]	1f[5]	3[2]	85

Fig.12 Example of Gradated Statistics

エントリに対し3.3 で述べた多次元binary trie を用いて多次元クラスタリングを施し，エントリを縮退操作に関して互いに独立なページに分割することによって，実効的な計算量を大きく減少させることを考えた。多次元trieは縮退操作とは逆に最上位ビットからその値が0か1かに従ってページ分割を行う。従って生成された各ページは各々全属性に対して上位ビットが相違することが保証され，下位ビットの縮退操作に関しては互いに独立となる。この多次元trieは，ページを問合せとみなして縮退統計表を検索する際のアクセスパスとして使用される。

4.6 動的ファイルに対するページ分割アルゴリズム

タプルの挿入／削除が頻繁に行われる所謂動的ファイルの環境ではページのオーバフロー／アンダフローが多発するが，先に述べた様にGKD-treeは再帰的なページ分割を行うクラスタリングのクラスに属しており，これに局所的，動的に対処することが出来る。

ページのオーバフローに対してはオーバフローページのページ分割により対処するが，ページのロードファクタを上げ，総ページ数を少なくする為にタプル数を2分する値を選ぶのが有利である。従ってここではページオーバフローの際のページ分割アルゴリズムとしては，k個の属性各々に対しタプル数を2分する値を求め，これらに対する$\frac{1}{2}$の値の最も小さいものを分割値とする。即ち，この場合分割値選択の自由度はkとなる。また，前述の様に最も頻繁に参照されるページは木の最深部に存在し，これが木の主要項となっていることから，このようなページのオーバフローの際には周辺のページをまとめてクラスタリングしなおすことが効果的である。

ページのアンダフローに対しては当該ページ及び内部ノードを抹消し，親ノードのポインタをアンダフローページの兄弟ページまたは兄弟木を指す様に値を更新することにより対処出来る。

また動的ファイル生成に対しては，タプル到着の初期の時点でタプルの分布を把握出来ず，Q(q)のpeakを横切るページ分割を行ってしまう場合がある（例えばQ(q)のpeak付近のタプルが初期に集中して到着する場合）。そこで動的ファイル生成に対しては初期に到着したタプルを数ページ分バッファリングし，充分なタプル分布情報を得た後にクラスタリングを行うことが有効と考えられる。

5. おわりに

データベースマシンＧＲＡＣＥの二次記憶系設計技法について考察した。提案する技法は一般化KD-tree (GKD-tree) による多次元クラスタリング技法であり，本技法はページ分割の際の分割属性，分割値選択に関する自由度を利用し，問合せ分布Q(q)，タプル分布D(t)を充分考慮したクラスタリングを実現し，従来の二次記憶系設計技法，更には多次元クラスタリング技法に比較し，大幅に平均アクセスページ数を減少させることが出来る。いくつかの例についてKD-tree との性能比較を行い，その性能を評価した。

一方で一様な低次元のpartial match query 等，リレーションをタプル方向に分割する限り平均アクセスページ数の減少をそれほど期待出来ない問合せ分布も存在する。この様な場合，リレーションの縦方向の分割（トランスポーズ，原空間の射影分解）とGKD-treeによるリレーションの横方向の分割を組み合せる技法が有用である。

今後，GKD-treeのコンパクション，探索の高速化技法等に加えて具体的な実装技法について更に検討を進めて行く予定である。

〔参考文献〕

1.Kitsuregawa,M. et al "Application of Hash to Data Base Machine and Its Architectue ", New Generation Computing, 1, 1983

2.Bentley,J,L. "Multidimensional Binary Search Trees Used for Associative Searching",CACM, Vol18(9),1975

3.Bentley,J,L. "Multidimensional Binary Search Trees in Database Applications", Trans. on Software Eng., Vol.SE-5(4), 1979

4.Chang,J.M.,and Fu,K.S. "A Dynamic Clstering Technique for Physical Database Design", Proc. of ACM SIGMOD Conf.,1980

5.Orenstein,J,A. "Multidimensional Tries Used for Associative Searching ", Info.Proc.Let.,Vol13(4), 1982

6.Tanaka,Y. "Adaptive Segmentation Schemes for Relational Files ", in Database Machines, Springer-Verlag, 1983

7.伏見他，"GRACE 二次記憶系に於ける拡張多次元クラスタリング技法"，情報処理学会第27回全国大会，1983

8.喜連川他，"パイプラインマージソータの構成"，電子通信学会論文誌，D-40，1983