情報学科CSコース情報システム（3年後期）

講義ノート
—第8回—

副次索引，転置ファイル，B木，
グリッドファイル，k-D木，シグニチャファイル

授業計画

- 第1回（10/03）情報検索(I) 適合率, 再現率, ベクトル空間モデル,類似検索（田中）
- 第2回（10/10）情報検索(II) tf/idf法，適合フィードバック，クラスタリング（田中）
- 第3回（10/17）情報検索(III) 情報検索の評価尺度（田中）
- 第4回（10/24）情報検索(IV) 協調フィルタリング, 推薦システム（田中）
- 第5回（11/07）情報システムの歴史：ハイパーテキストから Webサービスまで（田島）
  - Dexterモデル, Smalltalk, HyperCard, SGML, HTML, スタイルシート,XML, Xlink, SMIL, SOAP, REST, Ajax
- 第6回（11/14）XMLの基本，XMLのための問合せ言語（田島）
  - XPath，XQuery，XSLT，UnQL，各言語のパラダイムの違い
- 第7回（日程未定）XMLのためのスキーマ言語（田島）
  - DTD, XML Schema, RELAX NG, 各言語の表現能力の違い
- 第8回（11/28）副次索引（田中）
  - 転置ファイル，B木，グリッドファイル，k-D木，シグニチャファイル
- 第9回（12/05）空間アクセス法（田中）
  - Z-ordering, R木
- 第10回（12/12）マルチメディア情報検索（田中）画像検索, ビデオ動画像検索，Gemini
- 第11回（12/19） XMLの問合せ処理（田島）
  - 索引(DataGuide), Region Algebra, ノードラベリング方式, Join アルゴリズム, パス索引
- 第12回（12/26）Web 情報検索(I)：ランキング（田島）
  - PageRank, VisualRankなど
- 第13回（01/16）Web 情報検索(II)：コミュニティ発見と知識抽出（田島）
  - HITS，Webマイニング
- 第14回（01/23）Web 情報検索(III)：（田島）
- 第15回（01/30）試験

伝統的な索引法
Traditional Indexing Methods

検索の種類

- レコードの任意の属性値（の組合わせ）による検索
- Rivestの分類
  - ・完全一致質問：すべての属性値の指定
  - ・部分一致質問：一部の属性の値の指定
  - ・範囲質問：ある属性に関して値の範囲を指定
  - ・ブール質問：AND/OR/NOTの組み合わせ
- 近傍質問（nearest neighbor/best match query）
  salary≒45000 and age≒55

副次キー(Secondary Keys)と副次索引

主キー（Primary Key）による検索→B木やハッシュ

副次索引：転置ファイル(Inverted Files)

データベースの属性値による条件検索や，文書のフルテキスト検索の高速化

EMPLOYEE (name, salary, age)のsalary属性に関する転置ファイル

- 転置ファイルのレコードの論理的構造
  （salary属性値，この属性値を持つEMPLOYEEレコードへのポインター）
- 生成：RDBMS SQLの“CREATE INDEX”命令

フルテキスト検索：n-gram index

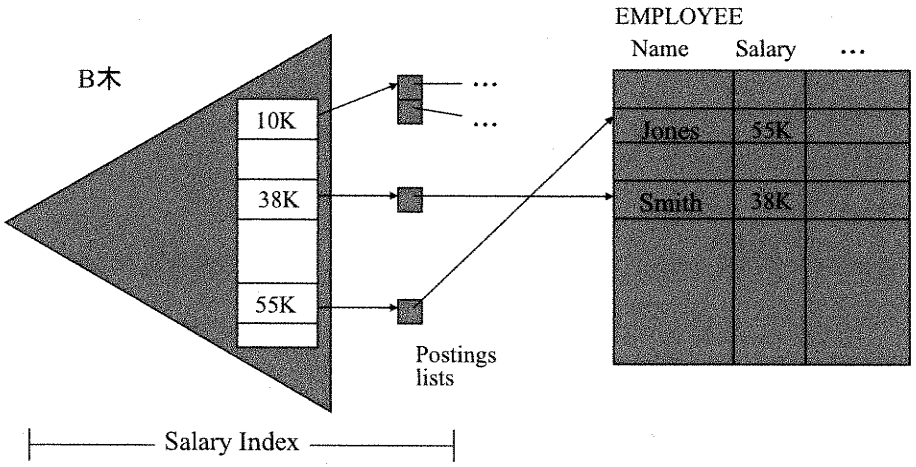
- 連続する2文字毎にインデックスを作成.
  ‘・・・東京都市場調査・・・’→ ‘東京’，‘京都’，‘都市’，・・

B木やハッシュ表による実装

ブール質問の処理：ポインターの集合操作

- salary=10K & age=24　　（salary, age転置ファイル）

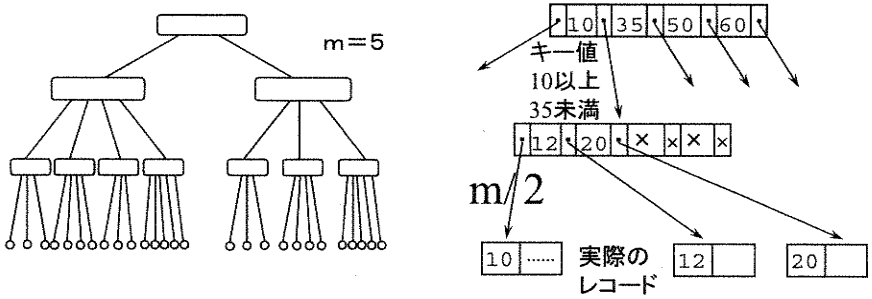
Salary属性に関するB木副次索引



B木（B-Tree）

m分木（m≧3），平衡木

- （1）各節点（葉以外）の子供の数の最大はmである。
- （2）各節点（葉以外）の子供の数の最小は「m/2⌉である（記号「x⌉は、 x以上の最小の整数、すなわちxの切り上げを意味する）。ただし、木根は例外とする。根の子供の数の最小は2である。
- （3）根から葉までの深さはどの葉についても同じである。



B木の節点構造

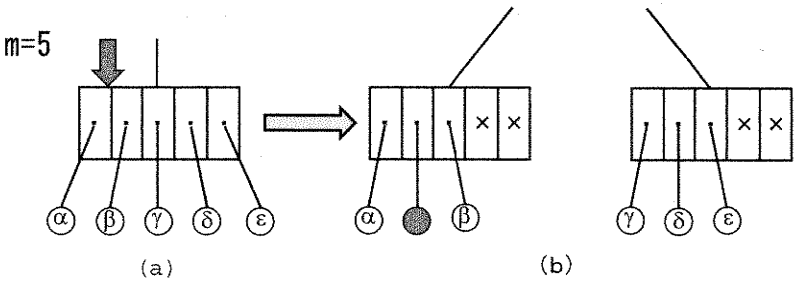
B木への挿入

(1)挿入位置の決定

(2)新しい葉節点の作成

(3)すぐ上の内部節点に新しいキー値とポインタを追加

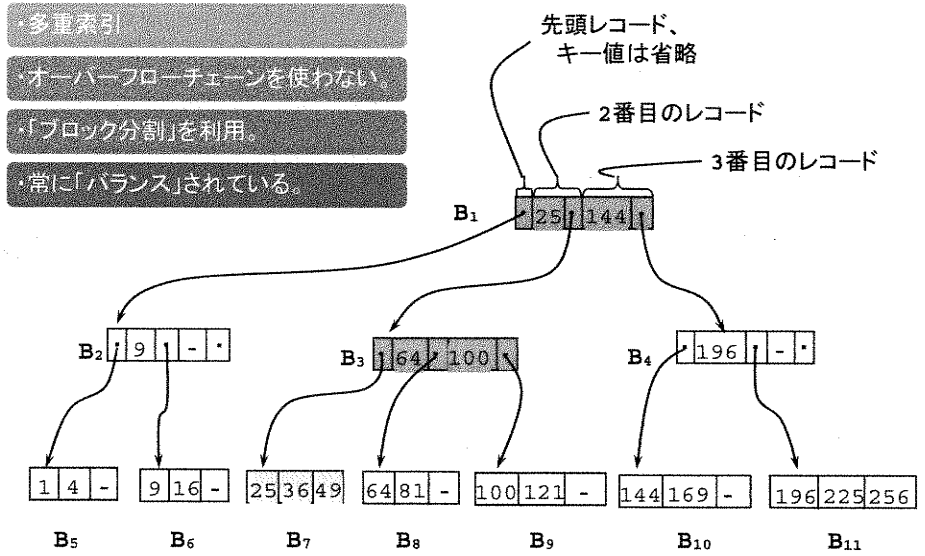
(4)内部節点があふれると，これを分割し，(3)へ



節点の分割

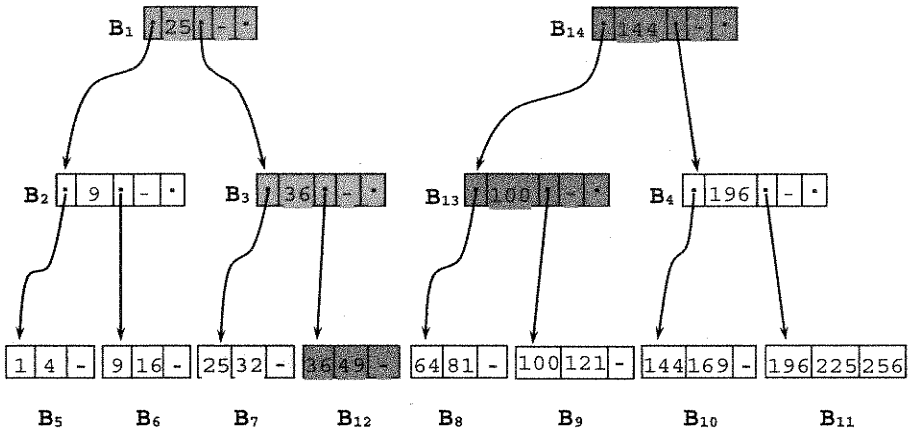
B木の例

- 多重索引
- オーバーフローチェーンを使わない。
- 「ブロック分割」を利用。
- 常に「バランス」されている。

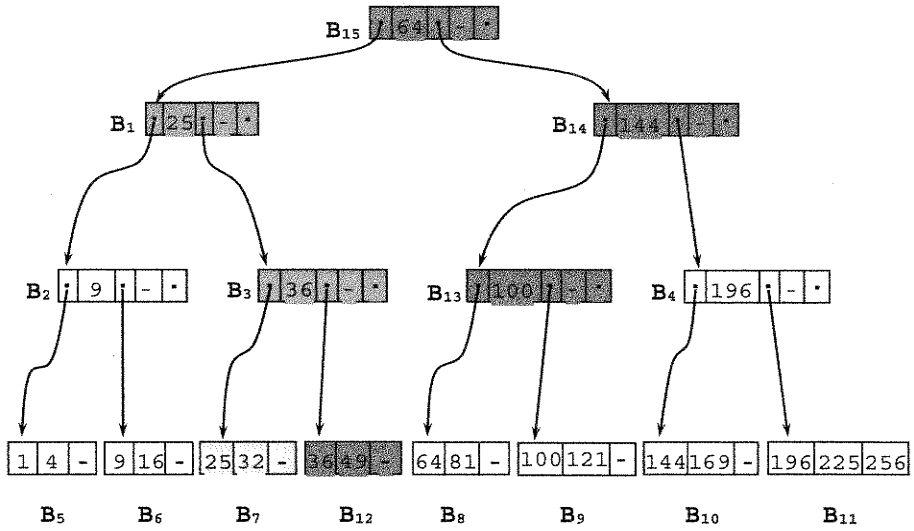


初期B木

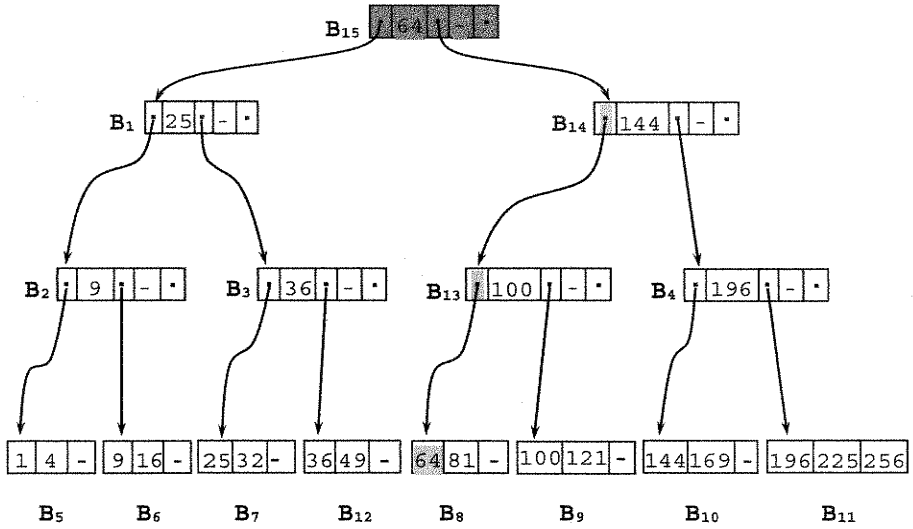
32の挿入プロセス



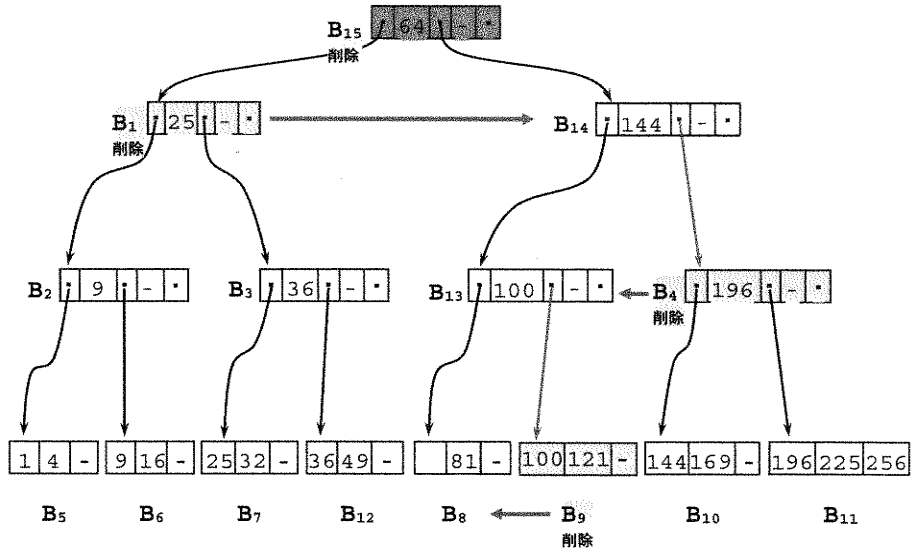
32を挿入後のB木



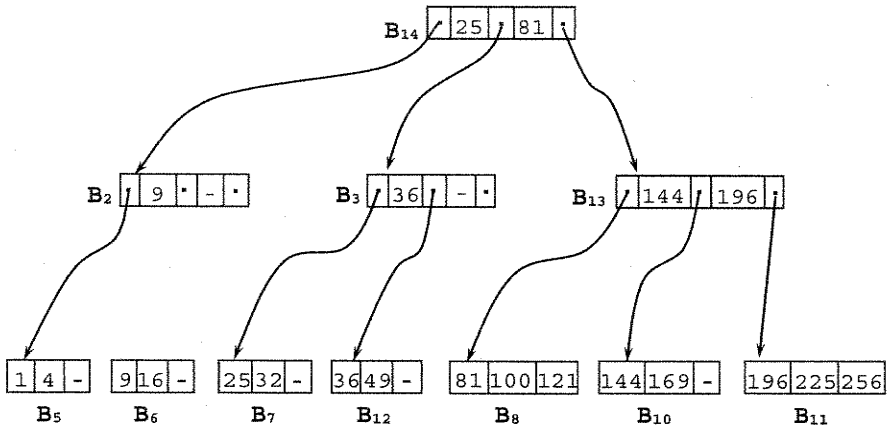
64を削除



64の削除による上位ノードへの波及



64を削除後のB木



B木の計算量

各節点： 最小限⌈m/2⌉本の枝

↓

木の高さ

log ⌈m/2⌉ nを越えない

↓

O(log n)

各節点内部の探索

高々mに比例

mを固定 → O(1)

m＝200

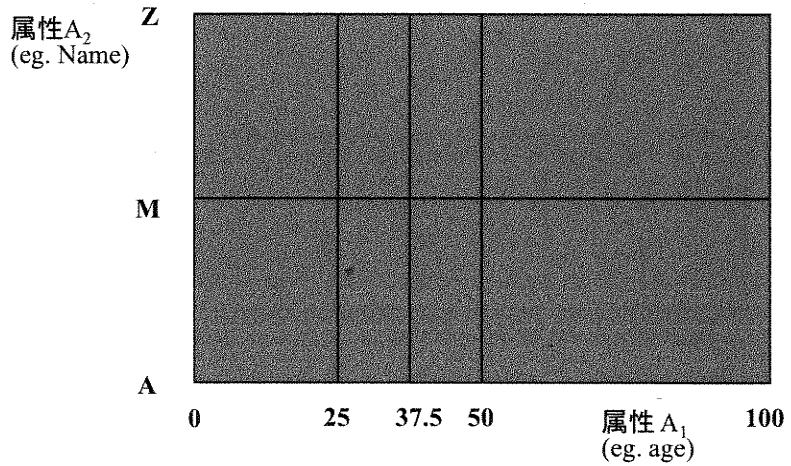
100万個までのデータ

4回のアクセス

副次索引：グリッドファイル（Grid File）

- k個の属性を持つレコードをk次元空間の点として表現
- k次元空間をグリッドに分割，各グリッドに点を均等配分
- 各グリッドは1つのディスクページに対応
- ディレクトリのレコード：
  （0≦age<25，“A”≦name<“M”，ディスクページ番号）

2つの属性を持つレコードのGrid File



副次索引：k-D木

元来、主記憶ベースのアクセス法

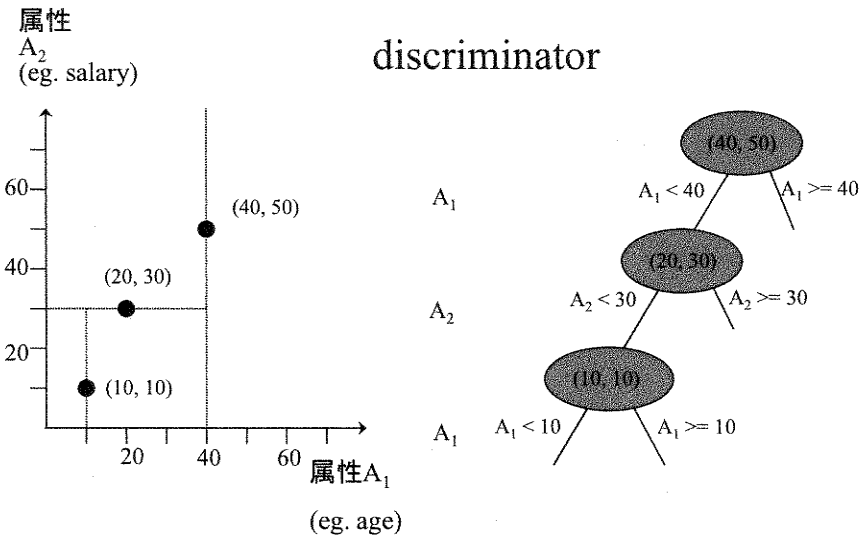
アドレス空間を重なりの無い領域に分割

構造は2分木（binary tree）

- 節点は（左ポインタ，データレコード，右ポインタ）
  2分木の各レベルにdiscriminator属性（Round robin法で決定）が対応

完全一致・範囲・近傍質問の処理に適する

k-D木の例



シグニチャファイル（Signature Files）

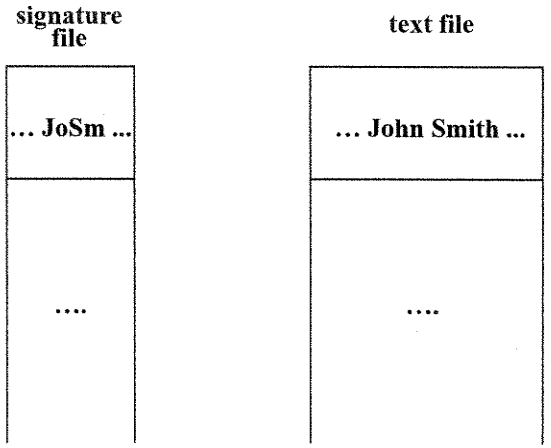
アイデア：“quick and dirty”フィルタ

- 答えにならない文書群のほとんどを早く排除する.
- フィルタの結果
  - 答えになる文書はすべて含む
  - 答えにならない文書（false drops）も若干含む
    全文テクストスキャニングなどで除く

シグニチャファイルの例

各文書中の各単語の先頭2文字を記憶

実際にはあまり用いられない



Superimposed Codingによる シグニチャファイル

Word	Signature
data	001 000 110 010
base	000 010 101 001
document signature	001 010 111 011

シグニチャ長: $F = 12$
各単語で1となるビット数: $m = 4$