コンピュータ概論B
ー ソフトウェアを中心に ー

#08 データベース

京都産業大学
安田豊

データベースとは

- 教科書 pp.103-
- 外見
  - データを決まった形式（フォーマット）で整理し蓄積したもの
  - レコード (Record) の存在
  - オブジェクト指向データベースのように決まったデータ型を用意しないタイプもある（例外は常にある）
- 目的
  - 入力・更新
  - 高速な検索、再利用

種類

- データモデルに適したタイプ
- カード型
  - 図書館蔵書カードのような一件一枚のもの
- ネットワーク(型)データベース
  - データの親子関係に注目
- リレーショナル(型)データベース
  - Relational Database
  - データの関係 (relation) に注目
  - 現在もっとも良く使われている
- 学生情報データベースを考える

カード型による学生情報データベース

- 一人一件
- 利点
  - 全ての情報がカードの中にあるのでカードを見つけられればあとの処理が簡単
- 欠点
  - 柔軟な検索が出来ない
  - キー以外の検索は一枚一枚繰ることに？
  - 通常はキーでソートして検索を容易にする
  - インデックス（複数）の利用も可能

探索法

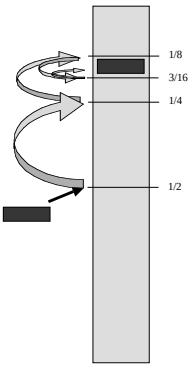
- より高速な検索のために
  - 高速とは？
  - CPU処理量(計算量)が少ない
  - ディスクアクセス量が少ない
- 多様な探索手法の存在
  - シーケンシャルアクセスとソート、二分探索
  - ランダムアクセスとハッシュ、インデクシング

シーケンシャルな探索

- 順次当たる方法 sequential
  - 単純総当たり
    - 図書カードをタイトルキーワードで繰る
  - ソート sort
    - 図書カードをタイトル順で並べておく
    - 妥当なところまでスキップ（調べるより送るだけの方がCPU処理量が少ない場合に有効）
- カード型データベースでは
  - 以下の記述は原理的な話として。ユーザインタフェイスとしてカード型に見せているだけで、内実は違う、という場合もありうる
  - 何か一通りの方法でのみソート可能
  - それ以外の方法でタイトル順のカードを作者で当たるときは総当たり

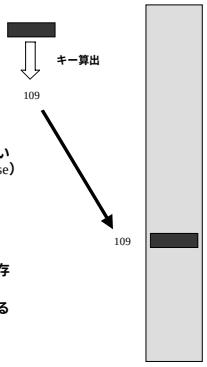
ランダムアクセスを利用した探索

- 二分探索 binary search
- 手法
  - sortされているカードの真ん中位置をまずアクセス
  - キーの大小から判断して、上下いずれのブロックに含まれるかを判定
  - 該当ブロックの真ん中を次にアクセス
- 利点：高速な検索 (log N)
- 欠点：順列のある場合だけ検索可能
  - 文字列部分マッチなどには使えない
  - ランダムアクセス可能なデバイス必須



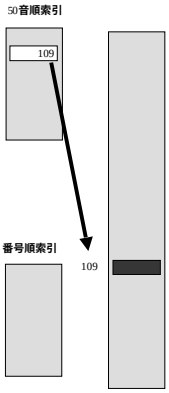
ランダムアクセスを利用した検索

- ハッシュ法 (hash)
- 手法
  - キーワードなどから何らかの数値を計算
  - 十分に拡散し、衝突が少なくなるように良い計算式を選ぶ（定式は逆に無く、case by case）
- 利点
  - うまくするとワンクッションだけでヒット
- 欠点
  - 重なったときの処理が面倒
  - ヒット率が入力データと計算式の相性に依存
  - データ領域の充填率が低い
  - 空きがあることを利用した高速化手法である



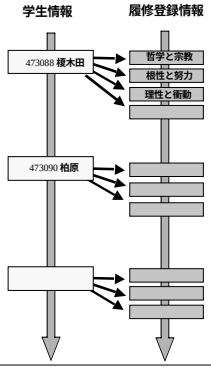
ランダムアクセスを利用した検索

- インデクシング (indexing)
- 索引(index)を利用する
  - 直接データを検索せずに
  - 検索に必要なデータだけをまとめた index を検索
  - そこにデータ位置が書かれている
  - 現実世界でも常用されている手段
- 利点
  - 複数のインデックスを持てる（名前順、学生番号順）
  - データの特性に依らず一般的に有効
- 欠点
  - インデックス作成に時間が掛かる（場合がある）
  - 追加より検索が圧倒的に多い場合に事前努力をする方式



ネットワーク型データベース

- データの親子関係に注目
- 利点
  - よく適合する用途には非常に高速（検索処理が実質不要）
- 欠点
  - 柔軟なデータ構成がとれない（設計時に完全に決定しておかなければならない）
  - 適合しない用途が後から現れても非常に効率が下がる
  - 例：三回生の多い履修科目はどれか？
- 銀行、業務システムなど変化の少ない用途には向いている



関係データベース

- 柔軟性とデータ独立性
  - ただ「関係」だけを表現
  - プログラムから独立したデータ表現
  - 後から項目の追加などが可能
- 数学的に完成したモデルを作った
  - Codd (1970, IBM) が理論的モデルを提唱
  - データを表組みで表現
  - 表と表の関係処理を集合演算モデルで定義
- RDB の完成
  - 1973 の SystemR (IBM), Ingress (UCB バークレー校)
  - 1979 Oracle
  - SQL の発明 (1986, ANSI 標準となる)

関係データベース

- 利点
  - 柔軟、プログラムとデータが独立
  - SQL という問い合わせ言語の便利さ
  - 数学的完全性
- 欠点
  - 概して低速
  - データ格納効率が高くならない
- 動かしながら開発したり将来変更が多そうなシステムに向く
- 現在もっとも多く市場で使われているタイプである
  - 欠点をマシン能力でカバーするという考え方
  - 現代的なソフトとハードの関係の典型例