関係演算アルゴリズムとコスト

- 関係代数演算を対象
  - 基本操作の記述
    - 関係データベース処理のアセンブラ
- コストの見積もり
  - 比較回数（CPUコスト）とディスクI/O
  - 関係 Rのサイズとして
    - R のタプル数を {R}
    - R のディスクページ数を |R|

関係データベース演算の種類(1)

- 1 入力演算 (Unary Operations)
  - 選択演算 (Selection)
  - 射影演算 ([Delta] Projection)
  - 集約演算 (Aggregation)
  - 整列 (Order-by)
  - 重複除去 (Duplication Elimination)
  - グループ化 (Group-by)

関係データベース演算の種類(2)

- 2 入力演算 (Binary Operations)
  - 結合演算 (Join)
  - 準結合演算 (Semi-Join)
  - 集合積演算 (Intersection)
  - 集合差演算 (Difference)
  - 集合和演算 (Union)
  - 直積演算 (Cartesian Product)
  - 商 (Division)

選択演算アルゴリズムとコスト

- 選択演算 (Selection)
  - Index なし
    - 比較回数: {R}
    - ディスク I/O 回数:|R|
  - Index あり
    - B-tree Index の比較回数、ディスク I/O:
      - Tree の深さ + 1: Tree の深さは $\log_F\{R\}$
    - ハッシュ Indexの比較回数、ディスク I/O:
      - 定数回

射影演算アルゴリズムとコスト

- 射影演算 (Projection)
  - ただし、重複除去は行なわない
    - デルタ射影:Delta Projection
  - Index の効果はなく、全スキャン
  - タプルアクセス回数:
    - {R}
  - ディスク I/O 回数:
    - |R|

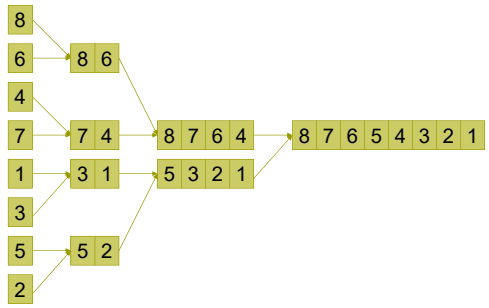
集約整列演算アルゴリズムとコスト

- 集約演算 (Aggregation)
  - Index の効果はなく、全スキャン
    - 比較回数: {R}
    - ディスク I/O 回数:|R|
- 整列演算 (Order-by)
  - メモリ上に乗る場合
    - 一般のソートアルゴリズム
      - クイックソート、マージソート、ヒープソート、etc

整列演算アルゴリズムとコスト

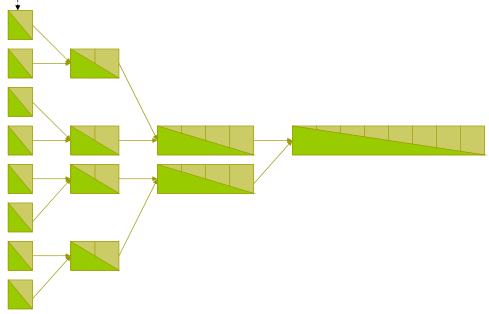
- メモリ上に乗らない場合の整列演算（外部ソート）
  - ディスクアクセスを前提とするとクイックソートは適さない
    - 最初に Logical Key で分割
    - → サイズが一定でない
  - マージソートが多用される傾向 ← 最初の分割は物理的
    - n-way マージした各段をディスクに格納
  - 2-way マージソートを前提にした場合のコスト
    - 比較の回数：$\{R\} \log_2 \{R\}$
      - ディスク I/O：2 $|R| \log_2 |R|$
      - 各段で読み出した後、結果を格納

2-way マージソート



2-way ページマージソート

A disk page



重複除去アルゴリズムとコスト(1)

- 重複除去 (Duplication Elimination)
  - 入れ子ループ (Nested Loop)
    - 1 つの対象に対して、同じ値のものを探す
    - 対象毎に全部をスキャン
    - 比較回数：{R}({R} - 1)/2
      - ({R} - 1)、({R} - 2)、…、1 の和
    - ディスク I/O：|R|(|R| + 1)/2
      - 最後に結果をディスクに書き込む
      - メモリが十分ある場合：|R|

重複除去アルゴリズムとコスト(2)

- ソート (Sort)利用
  - ソートした後で、頭から重複しているか見ていく
  - マージソートを前提（整列演算を参照）
    - 比較の回数：{R}(log{R}+1)
      - ソートのコストとその後の1回スキャンのコスト
    - ディスク I/O ：|R|(2 log|R|+1)
      - 各段で読み出した後、結果を格納
- ハッシュ (Hash)利用
  - 同じハッシュバケットの中で同じ値を除去
    - ハッシュコリジョンが無いとすれば 1 回スキャンで OK
  - 比較の回数：{R}
  - ディスク I/O： 2|R|

グループ化演算 (Group-by)（1）

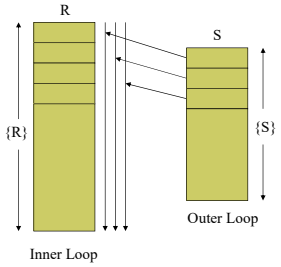
- 入れ子ループ (Nested Loop)
  - 1 つの対象に対して、グループになるものを探す
  - 対象毎に全部をスキャン（重複除去と同様）
  - 比較回数：
    - {R}({R} - 1)/2
      - ({R} - 1)、({R} - 2)、…、1 の和
  - ディスク I/O：
    - |R|(|R| + 1)/2
      - 最後に結果をディスクに書き込む

グループ化演算 (Group-by)（2）

- ソート (Sort)
  - ソートした後で、頭から見ていく
  - マージソートを前提（重複除去と同様）
    - 比較の回数：{R}(log{R}+1)
      - ソートのコストとその後の1回スキャンのコスト
    - ディスク I/O ：|R|(2 log|R|+1)
- ハッシュ (Hash)
  - 同じハッシュバケットの中でグループ化
  - ハッシュコリジョンが無いとすれば（重複除去と同様）
    - ハッシュ関数の適用、比較の回数：{R}
    - ディスク I/O： 2|R|

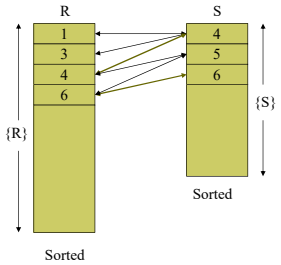
結合演算 (Join) （1）

- 入れ子ループ結合 (Nested Loop Join)
  - 全てのタプルの組み合わせの比較
  - 比較回数：{R} ×{S}
  - ディスク I/O：|R| × |S|



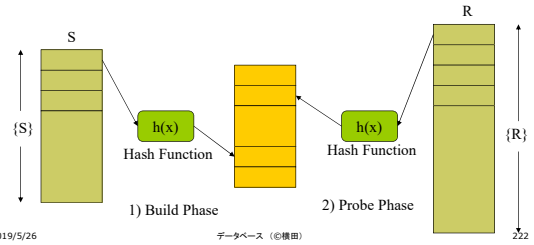
結合演算 (Join) （2）

- ソートマージ結合 (Sort Merge Join)
  - 先頭のタプル同士の比較
  - 比較の回数：{R}(log{R}+1) + {S}(log{S}+1) -1
  - ディスク I/O ： |R|(2 log|R|+1) + |S|(2 log|S|+1)



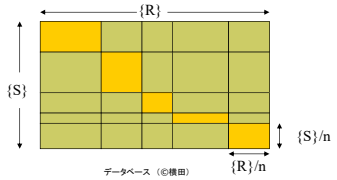
結合演算 (Join) （3）

- ハッシュ結合 (Hash Join)
  - 等結合のみ（θ 結合はだめ）
  - 両方の関係のタブルに同じハッシュ関数適用



結合演算 (Join) （4）

- R と S の全タプルがメモリ上に乗る場合
  - 比較の回数: {R}/n × {S}/n × n = ({R}×{S})/n
    - バケット数 n が十分大きい場合:{R}
  - ハッシュ関数の適用の回数:{R} + {S}



練習問題

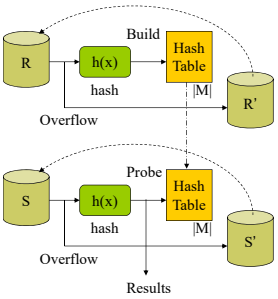
- 結合演算のコスト比較（イメージをもってもらうため）
  - R の Cardinality 1024
  - S の Cardinality 256
  - 全てメモリ上に乗るとして、
  - 入れ子ループ結合での比較回数は
  - ソートマージ結合での比較回数は
  - ハッシュ結合でRでProbe するとした場合の比較回数とハッシュ適用回数は

結合演算 (Join) （5）

- メモリに乗り切らない場合のハッシュ結合アルゴリズム
  - 単純ハッシュ結合 (Simple Hash Join)
  - GRACEハッシュ結合 (GRACE Hash Join)
  - ハイブリッドハッシュ結合 (Hybrid Hash Join)}
- 以下のコスト算出での前提
  - ハッシュテーブルに使えるメモリ量：|M|
  - 属性値は均等に分散
  - |R| > |S|

単純ハッシュ結合（1）

- ハッシュテーブルに入りきらない部分をディスクに格納
  - オーバーフロー分が無くなるまで繰り返す
  - R も S も（同じハッシュテーブルを使うため）
- 繰り返しの回数（期待値）：
  - L = |R|/|M| （≧ 1）



単純ハッシュ結合（2）

- CPUコスト：ハッシュ適用回数と比較回数
  - 1 回目
    - ハッシュ適用回数：{R} + {S}
    - 比較回数：{S}/L
  - i = 1,…, L 回目
    - ハッシュ適用回数：
      - {R} - (i -1)×{R}/L + {S} - (i - 1)×{S}/L
    - 比較回数：
      - {S}/L

単純ハッシュ結合（3）

- 全体としてのハッシュ適用回数:
  - L×({R}+{S}) – (L× (L-1))/2×({R}/L + {S}/L)
    = (L + 1)/2 ×({R} + {S})
    = (|R| + |M|) × ({R} + {S}) /2 |M|
- 全体としての比較回数:
  - ({S}/L) × L = {S}

単純ハッシュ結合（4）

- ディスクI/O
  - 1 回目
    - 読みだし:(|R| + |S|) 回
    - 書き込み:(|R| - |M| + |S| - |S|/L) 回
      - あるいは、|M| = |R|/L より(|R| +|S|) – (|R| + |S|)/L 回
  - 2 回目以降 i 回目
    - 読みだし:((|R| + |S|) - (i - 1) ×(|R| + |S|)/L 回
    - 書き込み:((|R| + |S|) - i ×（|R| + |S|)/L 回

単純ハッシュ結合（5）

- 全体（1 回目から L 回目までの和）
  - 2L × (|R| + |S|) - (L (L + 1)/2+(L (L - 1)/2 (|R| + |S|)/L
  - = L (|R| + |S|)
  - = |R| (|R| + |S|)/|M|