Database Systems : Mid2

Chapter 5

- Retrieving records when search key comprises of values from multiple sources of data.
- Geographical Information System
  - Query Types:
    - Partial Match
    - Range
    - Nearest Neighbour
    - Where-am-I
- Data Cubes
  - Typical storage example
  - Query Structure
- Multi-dimensional Queries
  - Points
  - Rectangles
- Cube queries implementation
- Limitations of Conventional Methods:
  - Range queries strategy
  - Nearest Neighbour Limitations
  - Multiple dimension queries and sorting on attribute

- **Hash Like Approach**:
  - GRID Lines
  - Partitioned Hash Function
- GRID lines :
  - Concept
  - Storage : Indirection and Buckets
  - Insert, Look-up and Add operations
  - Performance for various queries (*Range, NN*)
  - Perks [1]
  - Demerits[2]
- Partitioned Hash Functions
  - ***Concept***
- Comparison of GRID file and Partitioned Hash Function
  - PSH useless for NN
  - PSH has better memory utilisation

        - PSH optimim for partial match queries
    - Handling Tiny Buckets : Block headers and and splitting block on addition

- **Tree Approach:**
    - Multiple Key Indexes
    - Kd Trees
    - Quad Trees
    - R-Trees
- Multiple Key Indexes:
    - Indexing for each attribute in a tree-like manner
    - Performance for different cases
- kD Tree:
    - Concept
    - Operations and Lookup (*Binary Search*)
- Quad Tree:
    - Concept
- R Tree:
    - Concept
- **Bit-map Indexes:**
    - Concept
    - Application
    - Compression

Chapter 6

- Major parts of Query compilation:
    - Parsing
    - Rewrite
    - Physical plan generation
    - *Last two steps called query optimisation*

- **Operations:**
    - Union, Intersection and Differences
      UNION, INTERSECT, EXCEPT
      What each operator does.
      Requirement : Schema of argument relations much be identical
      ******BAG vs SET******
      UNION ALL for BAG
    - Selection

    WHERE
    Takes a Relation and a Condition as argument.
    Condition can be Arithmetic, Boolean or Comparison.
  - Projection
    SELECT
    Can take:
    A single attribute of R or a list
    x —> y, to rename an attribute x of R to y
    E —> y, to rename an expression E to y
  - Product
    Cartesian product formation, FROM, WHERE, SELECT
  - Joins
    JOIN
    Natural Join
    Theta (*any op other than =*) Join
    Equi join
    Outer Join
    Left Outer Join
    Right Outer Join
  - Duplicate Elimination
    DISTINCT
  - Grouping
    GROUP BY
  - Sorting
    ORDER BY
    Tau is the Sort by operator, takes a list of attributes, in order to sort

- Expression Tree
- Scanning Tables
  - *Table Scan*
  - *Index Scan*
- Sorting while scanning
  - BTree Read
  - Main Memory fit
  - Too large, then multi-way merging
- Notations:
  - B(R)
  - T(R)
  - V(R,a)
  - M

- Time slide
- Iterator
  - Open
  - GetNext
  - Close
- Operators:
  - Unary
  - Full Relation (need full view of R)
  - Full Relation, Binary

- **One Pass Algorithm**
  - Unary, Complexity
  - DISTINCT, Complexity
  - Grouping, Complexity
  - Union, process, Complexity
  - Intersection, process, Complexity
  - Difference, process, Complexity
  - Bag Intersection, process, Complexity
  - Bag Difference, process, Complexity
  - Product
  - Natural Join, Process , Complexity = B(R)+B(S)
  - Nested Loop Join, Complexity tuple-wise= T(R)*T(S)
  - Nested Loop Join, Complexity tuple-wise= B(R)*B(S)/M , Derivation
  - **TABLE IN SLIDES**

- **Two Pass Algorithms : Sort Based**
  - Distinct : C = 3*B(R), M = root(B(R)), B < M^2 required
  - Same for grouping and aggregation
  - Union, Intersection and Bag : C = 3*(B(R)+B(S)), required : (B(R)+B(S) < M^2
  - Simple Sort based Join : C = 5*(B(R)+B(S)), required : B(R)< M^2 and B(S) < M^2
  - Modification for worst case, Simple Sort based Join
  - Sort-Merge-Join C = 3*(B(R)+B(S)), required : B(R)+B(S) < M^2
  - Worst case, use Nested Loop Join

- **Two Pass Algorithm based Hashing**
  - Basic Idea
  - Partitioning Relation

    - Distinct, required : $B(R) < M^2$
    - Grouping Aggregation, $C = 3*B(R)$, $M = \text{root}(B(R))$
    - Binary : $C = 3*(B(R)+B(S))$, required : $B(R)+B(S) < M^2$
    - Join : One pass of corresponding buckets : $C = 3*(B(R)+B(S))$, required : $\min(B(R),B(S)) < M^2$
    - Hybrid Hash Join : $C = (3 - (2M/B(S)))(B(R)+B(S))$

- Sort based vs Hash Based
    - Size
    - Sort Order
    - Bucket Size
    - Writing sub-lists

- **Index based Algorithms**
    - Clustered relations and index
    - For clustered, $C = B(R)/V(R,a)$
    - For non-clustered, $C = T(R)/V(R,a)$
    - Organisation:
        - Clustered File Organisation
        - Clustered Relation
        - Clustered Index
    - Join : $C = T(R)*T(S)/V(S,Y)$
    - Advantages of Index Join:
        - R is small, and V large, thus C is low.
        - If select before join, then most tuples not examined.
    - Join using sorted index

- **Buffer Management**
    - Purpose delay minimisation
    - Control main and virtual memory, target to avoid thrashing
    - To clear buffer when full
        - LRU
        - FIFO
        - Clock : upon read, use set to 1, manager removes 0, sets 1 to 0
        - System Control
    - Pinned Blocks, manager avoids

- Sort based Algos allow change of M.
- Hash based allow change in number of buckets as long as it fits in main memory

- **Parallel Algorithms**
  - Modes:
    - Shared memory
    - Shared Disk
    - Shared Nothing
  - Shared nothing has tuple-at-a-time operations in parallel
  - Hash used to distribute operations
  - No. Of disk I/Os remain constant, time reduces
  - Time = 1/processor of time + cost of shipping

Chapter 7

- SQL Query -> Expression Tree -> Logical Plan Tree -> Physical Plan Tree
- Steps:
  - Parsing
  - Rewrite
  - Physical Plan Generation
  - *Last two called **Query Optimiser***
- Query -> Parser -> Pre-processor -> LQ Plan Generator -> Rewriter -> Preferred LQ Plan
- **Parse Tree** - Nodes are atoms or syntactic categories
- Pre-processor - Semantic checker
  - Relation use
  - Attribute use
  - Types
- Commutativity, Associativity used used to improve query plans
- Pushing Selections - Reduces no. of tuples **LAWS**
- Pushing Projections - **LAWS**
- Pushing Distinct - **LAWS**
- Pushing Aggregation and Grouping - **LAWS**
- **Parse Tree to LQ Tree**
  - Replaces nodes with Relational Algebra terms
  - Simplify plan
  - **Including Conditions through *two argument selection and next equate selection***

- **Estimating Cost of Operations**
  - Estimating Projection
  - Estimating Selection

- Inequality, assumption 1/3
- Estimate of Join : C = T(R)*T(S) / max (V(R), V(S))
- T(R)T(S)/max(V(R,y1), V(S,y1)) max (V(R,y2), V(S,y2)), for multiple attributes
- Estimation of other operations

- **Cost-based Plan Selection**
  - Incremental Computation of Statistics
    - T(R)+1
  - Approaches:
    - Top Down
    - Bottom Up : Selinger Style
  - Greedy Heuristic
  - Brach and Bound
  - Hill Climbing
  - Dynamic Programming
  - Selinger-style Optimisation

- **Choosing the Order of Joins**
  - Join Tree
  - **Left Deep Join Tree : Best, 2 reasons** [Limit permutations, Fit well with algos]
  - Right Deep Join Tree
  - Bushy Join Tree
  - Dynamic:
    - Consider All
    - Subset
    - Heuristic
  - Seliger style Optimisation
  - Greedy Algorithm

- **Physical Query Plan Selection**
  - Selection
  - Join
  - Materialisation vs Pipeline
  - Selection, Projection excellent for pipelining