Distributed Databases Review

CS347

June 6, 2001

1

Fragmentation

- How to partition relation into various pieces/fragments
- Types:
  - Primary Horizontal
  - Derived Horizontal
  - Vertical
  - Hybrid of the above possible
- Desiderata
  - Completeness: don't lose tuples
  - Disjointness: no duplicate tuples
  - Reconstruction: make sure you can get back original relation

2

Minterm-based horizontal frag.

- Simple predicates $P_r = \{p_1, p_2, .., p_m\}$ and R.
- Generate "minterm" predicates from P_r
- Eliminate and simplify (depends on app semantics)
- Generate fragment $\sigma_m(R)$ for each minterm "m".

- Simple predicates:
  - P_r must be complete (do not under fragment) and minimal (do not over fragment)
  - Use predicates occurring in most frequent queries

3

Derived Horizontal

- R fragmented into $\{R_1, R_2, \ldots, R_n\}$
- For S, derive $\{S_1, S_2, \ldots, S_n\}$ where $S_i = S \ltimes R_i$

- Useful for join queries between R and S
- For completeness: referential integrity constraint $S \rightarrow R$
- For disjointness: join attribute is key of R

Vertical

- Split R by attributes
- Repeat key attribute in each vertical fragment
- Attribute affinities define grouping

4

Localization

- Convert query tree on relations into query tree on fragments
- Simplify ($\cup$ up & π,σ down)
- Rules
  - [R: False] $\Rightarrow \emptyset$
  - $\sigma_{C1}[R: C_2] \Rightarrow [R: C_1 \wedge C_2]$
  - $[R: C_1] \underset{A}{\bowtie} [S: C_2] \Rightarrow [R \underset{A}{\bowtie} S: C_1 \wedge C_2 \wedge R.A = S.A]$

  - Give vertical fragments $R_i = \Pi_{Ai}(R)$, for any $B \subseteq A$:

    $\Pi_B(R) = \Pi_B [\underset{i}{\bowtie} R_i \mid B \cap A_i \neq \emptyset]$

5

Distributed Operators

- **Sort**
  - Basic sort (sort each individual fragment)
  - Range partitioning sort (partition by sort attribute + basic sort)
  - Parallel external sort merge (local sort + range partition by sort attribute)
  - Key issue: selecting partitioning vector
- **Join**
  - Partitioned join (only for equi-joins)
  - Asymmetric fragment+replicate join (fragment R, replicate S)
  - General fragment+replicate join (fragment and replicate R and S, join all possible pairs)
  - Semi join programs (to reduce communication cost)

6

Query Optimization

- Exhaustive + pruning
  - Enumerate all possible QEP's with given set of operators
  - Prune using heuristics (e.g., avoid cartesian products)
  - Choose minimum cost QEP
- Hill climbing
  - Initial feasible QEP + set of QEP transformations
  - Iterate until no more cost reduction
    - Transform current QEP all possible ways
    - Check cost of each transformed QEP
    - Choose minimum and set as current QEP

7