Picking Up Where the SQL Optimizer Leaves Off

Howard Schreier, U.S. Dept. of Commerce, Washington DC

· Case study, matching spatial data

· Diagram (two dimensions)

· SQL code (three dimensions)

· Scale is the issue

Almost 20 **trillion** rows in intermediate result

· Parallel partitioning

Divide into pieces; solve separately; makes task no smaller, but more manageable

· Variant: Upper bound on distance

- Size of join unchanged
- Smaller intermediate result passed to “back end”
- Wrong answer (but subset of correct answer)

· Anticipatory subsetting

- No change in results, since restriction is redundant
- Reduction in size of join (maybe)

· Redundant equi-join condition

No change in results; avoid complete brute-force evaluation

· Building block code

Based on variant with upper bound; incorporates

- anticipatory subsetting
- redundant equi-join condition

· Performance

3:50 using brute force; 2:50 with anticipatory subsetting; 0:33 with redundant equi-join condition (only); 0:25 with both techniques

· Review

- Original problem has simple SQL solution
- That solution does not scale well
- Variant problem
  - Returns only a subset of the correct result
  - Does permit performance tuning

· Concentric partitioning

Set upper bound; solve using building block code (result is subset of the correct result); remove points successfully matched; increase upper bound; repeat as many times as necessary

· Choosing radii

- Too small: no match-ups

- Too large: inadequate performance gain over the naive code

· Strategy overview

- Apply parallel partitioning
- Use concentric partitioning to solve each piece
- Each iteration uses the building block SQL code (which incorporates anticipatory subsetting and redundant equi-join condition)

· Equi-join methods in PROC SQL

Sequential; index-based; Hashing

· Providing memory for hashing

Partition; use BUFFERSIZE=

· Data for testing

Synthetic; full scale; realistic

· SAS coding

· Decisions

- Number and shape of parallel partitions
- Progression of radii

· Objectives

- Small joins (to enable hashing)
- Small result sets from joins (to manage disk footprint)
- Avoid excessive overhead
- Limit aggregate number of pairings considered (<< 20 trillion)

· Parallel partitions

· Concentric partitions: radii

· Results

- All points matched
- Building block code run 926 times
- Aggregate 62 hours
- 50 parallel partitions
  - Minimum: 13 minutes
  - Maximum: 8 hours
- Aggregate joins: approximately 39 trillion
- Estimated 70 billion rows passed to “back end”