The value of merge join and hash join in Microsoft SQL Server and relational query processing

Goetz Graefe
Microsoft SQL Server
Why this study?

• Blasgen & Eswaran – 20 years ago
  Merge join & (index) nested loops cover all cases pretty well
• DeWitt, Sacco, others – 10-15 years ago
  Hash join is great for large unsorted inputs
• Analytical studies, simulation, experiments
Success without merge/hash join

- Sybase & Microsoft SQL Server
  Until recently used only nested loops
  Successful for over 10 years!
  Even used in data warehousing!

- Focus on OLTP
  Sybase invented stored procedures
  Microsoft leads SMP TPC-C efficiency

- Focus on canned reports
  Perfectly possible with tuned index sets
Are the prior studies wrong?

- Small evaluation sets
  - Few tables, few queries
- Insufficient credit to index tuning
  - Fixed set of indexes
- This study:
  - Still limited yet non-trivial queries & tables
  - Indexes tuned using a “tuning wizard” tool
    - Large set of possible indexes, integrated with query optimizer
- Next study
  - Indexes tuned specifically for available algorithms
SQL Server 7.0 query processor

- Nested loops with stored or temporary indexes
- Merge join & hash join (incl. hash teams)
- Index intersection, union, difference, & join
- Star joins: star indexes, cross-product, & semi-join reduction
- Constraints exploited for selectivity estimation & cost calculation & query simplification
- Parallelism on SMPs
- Content queries (“contains”, “near”, “about”)
- Optimized update plans (indexes, constraints)
- Heterogeneous & distributed queries
Relevant SQL Server tools

- Graphical show plan
- Profiler
  Captures workloads & events (e.g., deadlocks)
  Filters on application, database, user, operation, elapsed time, etc.
- Index tuning wizard
  Optimizes a workload captured with the profiler
  Reconsider all indexes – only add indexes
  Increase / decrease database size
  Uses query optimizer to assess choices
Experimental setup

• TPC-D database
  scale factor = 1 (1 GB raw data)

• Old & new TPC queries
  22 queries total

• Flags to disable
  Index join, merge join, hash join, hash teams
  Stream aggregation, hash aggregation

• Indexes in simple database design
  Primary keys, foreign keys, dates
Performance with simple indexes

Simple indexes

Time [% of entire NL run]

Queries & Algorithms

September 22, 1999
Performance with simple indexes

- NL=MJ >> HJ=All: #1, #15
  Hashing improves performance
  Aggregation, not join, make the difference
  Early aggregation missing in sort code

- NL=MJ=HJ=All: #2, #13, #16, #17
  No really meaningful difference
  Indexes are sufficient to select & retrieve rows

- NL > MJ > HJ=All: #3, #5, #7, #8, #9, #11

- NL >> MJ=HJ=All: #4, #14, #19
  Need some method for large unindexed inputs
Workload performance

Time [% of NL run]

Entire workload

Workload performance
Workload performance

• Only NLJ is not competitive
  Due to simplistic index design
• Hash-based query processor performs best
• NLJ + MJ are very competitive
  40% difference to full QP with hash join
  That’s 9 month of hardware improvements
  • Presuming 2x CPU speed in 18 months
Poor indexing strongly favors hash join
Blasgen & Eswaran were right all along …?
Tuned index set

Tuning wizard retains primary keys indexes
  • 7 indexes on *line item*, up to 7 columns
    Total 26 columns indexes
  • 4 indexes on *orders*, lots of redundant keys
  • 2 indexes on *part supply*
Performance with tuned indexes

Tuned indexes

Queries & Algorithms

Time [% of NL run on simple indexes]
Performance with tuned indexes

• Overall performance improvements
  Except queries 6, 12, 19
  Tuning wizard minimizes workload time
    • Not the time for each individual query

• More queries in these patterns
  NL > MJ=HJ=All
  NL=MJ=HJ=All
Workload performance

Entire workload, tuned indexes

Time [% of NL run on simple indexes]
Workload performance

• All algorithm combinations are fast
  Maximal difference 45 vs. 20, or 21 months
• Either MJ or HJ serve well
  Having both adds 20% performance – 5 months
Conclusions

• Either indexing or merge / hash join
• Are hash join & merge join just an excuse for poor (non-automatic) indexes?

• Next steps
  Tune & analyze for specific algorithms
  Analyze bitmap operations & star joins
  Look for orders of magnitude – multiple years
  • Pre-computed query result – indexed views
  • Fully automatic indexing & tuning
  • Caching data & query results on desktops
More information

• [www.microsoft.com/sql](http://www.microsoft.com/sql)
• Msdn.microsoft.com
• Technet.microsoft.com
• Research.microsoft.com