Recent Advances in Query Optimization

Tutorial by:

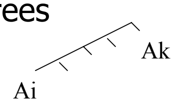
S. Sudarshan
IIT Bombay
sudarsha@cse.iitb.ernet.in
www.cse.iitb.ernet.in/~sudarsha

Talk Outline

- System R, Volcano
- Recent extensions (including OODBs, ORDBs)
- OLAP
- Materialized views:
  - maintenance, use and selection, continuous queries
- Caching of Query Results
- Data Warehouses and Virtual Warehouses

System R

- Join order selection
  - A1 ⋈ A2 ⋈ A3 ⋈ .. ⋈ An
  - Left deep join trees



  - Dynamic programming
    - Best plan computed for each subset of relations
      - Best plan (A1, .., An) = min cost plan of(
        A1 ⋈ Best plan(A2, .., An)
        A2 ⋈ Best plan(A1, A3, .., An)
        ….
        An ⋈ Best plan(A1, .., An-1))

System R (cont)

- Selects and projects pushed down to lowest possible place
- Sort order
  - join may be cheaper if inputs are sorted on join attr
  - => Best plan(set-of-relations, sort-order)
- Starburst (successor to System R)
  - retains single query block-at-a-time cost based optimization
  - + heuristic Query Rewrite
    - including decorrelation of nested queries

Decorrelation

- Idea: convert nested subqueries to joins
- Consider

  select * from emp E
  where E.numchildren <>
  (select count(*) from person
  where person.parent = E.name
- Can't always express using basic rel. algebra
- Long history:
  - special cases: Kim 88, Dayal 88, Muralikrishna 93
  - general case: P. Seshadri et al 95: use outerjoin

Decorrelation (cont)

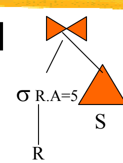
- Pushing semijoins into decorrelated query
  - use selections on correlation variables
    - select * from R, S
      where R.A = S.A and R.B = (select min(T.B)
      from T where T.A=R.A)
    - don't evaluate groupby/min on all of T:
    - GB $_{T.A,\ min(T.B)}$ (T SJ $_{T.A=R.A}$ (R $\bowtie_{R.A=S.A}$ S)

Magic Rewriting

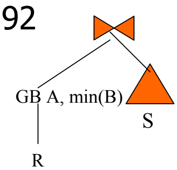
- Recursive views are now part of SQL-3, supported by DB2 and Oracle already
- Magic rewriting pushes semijoins through recursive views
  - path (X, Y) :- edge (X, Y)
    path (X, Y) :- edge (X, Z), path(Z, Y)
    Query: ?path(Pune, Y)
- Long history, see survey by Ramakrishnan and Ullman

Predicate Movearound

- Idea: pull R.A=5 up, infer S.A=5, and push S.A=5 down into subtree S



- Generalizes to any constraints
- History:
  - Fold/unfold transformation in logic programs
  - Aggregate constraints and relevance RS, VLDB91
  - Fold/unfold and constraints RS, ILPS 92
  - for SQL LMSS, SIGMOD 93
- Aggregate constraints

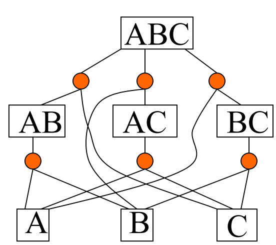


Volcano Extensible Query Optimizer Generator

- General purpose cost based query optimizer, based on equivalence rules on algebras
  - eg equivalences: join associativity, select push down, aggregate push down, etc
  - extensible: new operations and equivalences can be easily added
  - notion of physical properties generalizes “interesting sort order” idea of System R
  - Developed by Graefe and McKenna 1993
- Follow up to EXODUS, but much more efficient

Key Ideas in Volcano

- DAG representation of query
  - Equivalence node ☐ and operation nodes ●
  - Compactly represents set of all evaluation plans
    - choose one child of each equivalence node, and all children of operation nodes



Key Ideas of Volcano (Cont)

- Hashing scheme used to efficiently detect duplicate expressions
  - gives ID to each equivalence node, hash function of operation nodes based on Ids of child equivalence nodes
- Physical algebra also represented by DAG
- Best plan found for each equivalence node
  - use cheapest of child operation nodes
  - dynamic programming: cache best plans
  - branch and bound pruning used when searching

Main Benefits of Volcano

- Highly Extensible
  - can handle arbitrary algebraic expressions
  - new operators and equivalence rules easy to add
    - must be careful of search space though
- Yet (reasonably) efficient
  - generalizes the dynamic programming idea of System-R optimizer
  - Optimizations of Pellenkroft et al. [VLDB 97] eliminate redundant derivations for joins
- Ideas are used in MS SQL Server and Tandem

Parametrized Query Optimization

- Some parameters to the query may not be available at optimization time
  - selection constants (e.g. in stored procedures)
  - memory size
- Idea:
  - come up with a set of plans optimal at different points in parameter space,
  - select best when parameters are known at run time
- Work in this area
  - Ganguly [VLDB 1998], Ganguly and Krishnamurthy [COMAD 95], Ng et al [SIGMOD 92]

Parametric Query Opt (Cont)

- Results of Ganguly [1998]
  - Number of parametrically optimal queries is quite small, so idea is practical
  - nice algorithms for single parameter case
  - extended above to two parameter case, but general case is harder
- Optimization for best expected case (P. Seshadri, PODS 99)

Sampling and Approximate Query Answering

- In databases, sampling originally proposed for query size estimation (estimate need not be perfect) Li and Naughton [94], Olken [93]
- Used today for generating quick and dirty (fast but approximate) results
  - especially for aggregates on large tables
- Online aggregates (Hellerstein ..)
- Generating histograms (Ioannidis ..)

Optimization in OODB/ORDBs

- Major issues
  - Path expressions:
    - e.g. forall (p in person) print (p->spouse->name)
    - can convert pointer dereferences to joins
    - can "assemble objects" in a clever sequence to minimize I/O (Graefe 93, Blakeley et al, Open OODB optimizer 95)
  - Path indices
    - e.g. forall (p in person suchthat p->spouse->name = "Rabri") …

Optimization in ORDBs

- Expensive predicates/functions in selects/projects
  - e.g. selects based on image manipulation
  - usual heuristic of "push select predicates to lowest possible level" does not work
  - Hack to System R: treat predicates like joins
    - not an issue with Volcano
    - also heuristics to limit search space (Hellerstein and Naughton (93,94), Chaudhuri et al (93)

Extended ADTs

- ADTs are a simple way to add new types to a database. Used extensively in data blades/cartridges/…
- Extended ADTs -- understand some semantics of ADT functions, and optimize
  - e.g. if Image.smooth().clip(10,10) is equivalent to Image.clip(10,10).smooth choose the one that is cheaper to compute
  - Predator ORDB supports such optimizations (P. Seshadri [1998])

Multi Query Optimization

- Idea: Given a set of queries to evaluate, exploit common subexpressions by materializing and sharing them
- Problems: Many equivalent forms of a query
  - Some have CSE, others dont. E.g.:
    - $R \bowtie S \bowtie T$ and $R \bowtie P \bowtie S$ versus
    - $R \bowtie S \bowtie T$ and $R \bowtie S \bowtie P$
- Exhaustive algos: Sellis [1988], and others
  - try every combination of forms of every query.
  - problem: cost is doubly exponential

Multi Query Optimization (Cont)

- Heuristics
  - Find best plans for each query, look for CSEs in best plans
    - Subramaniam and Venkataraman [SIGMOD98]
    - Volcano SH [RSSB99]
  - When optimizing query i, treat subparts of plans for earlier queries as available cheaply
    - Volcano RU [RSSB99]

Greedy Heuristics for MQO

- Greedy heuristic:
  - Repeat
    - find subexpression which if materialized and shared will give most benefit (cheapest plan)
      - subproblem: given some subexpressions are materialized, find best plans for given queries
      - also: update the best plans ***incrementally*** as new subexpressions are checked for materialization
    - materialize above subexpression
  - Until no further benefits can be got

Greedy Heuristic (Cont)

- Monotonicity addition to greedy heuristic:
  - Benefit of materializing a subexpression cannot increase as other subexpressions are materialized
  - Assume above, and keep heap of overestimates of benefits -- reduces number of benefit recomputations
- Performance study shows greedy heuristic gives very significant benefits on TPCD queries at reasonable cost
- Volcano-SH and Volcano-RU are very fast but give much less benefits than Greedy

OLAP - Data Cube

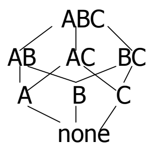
- Idea: analysts need to group data in many different ways
  - eg. Sales(region, product, prodtype, prodstyle, date, saleamount)
  - saleamount is a measure attribute, rest are dimension attributes
  - groupby every subset of the other attributes
    - precompute above to give online response
  - Also: hierarchies on attributes: date -> weekday, date -> month -> quarter -> year

OLAP Issues

- MOLAP: cube in memory, multi-dimensional array
- ROLAP: cube in DB, represented as a relation
-

Type	Size	Colour	Amount
Shirt	14	Blue	10
Shirt	20	Blue	25
Shirt	ALL	Blue	35
Shirt	14	Red	3
Shirt	20	Red	7
Shirt	ALL	Red	10
Shirt	ALL	ALL	45
…	…	…	…
ALL	ALL	ALL	1290

Data Cube Lattice

- Cube lattice
  - 

- Can materialize some groupbys, compute others on demand
- Question: which groupbys to materialze?
- Question: what indices to create
- Question: how to organize data (chunks, etc)

Cube: Selecting what to materialize

- Basic cube: materializes everyting
- Greedy Algo: max benefit per unit space
  - benefit computation takes into account what is already materialized
  - Harinarayanan et al [SIGMOD 96], Gupta [ICDE97], Labio et al …
- Smallest Algo
  - Deshpande et al [SIGMOD 98]

Materialized Views

- Can materialize (precompute and store) views to speed up queries
  - Incremental maintenance
    - when database is updated, propagate updates to materialized view
  - Deciding when to use materialized views
    - even if query does not refer to materialized view, optimizer can figure out it can be used
  - Deciding what to materialize
    - based on workload, choose best set of views to materialize, subject to space constraints

Incremental View Maintenance

- E.g. $R \bowtie S$

  $(R \cup i_r) \bowtie S = R \bowtie S \cup i_r \bowtie S$

  $(R - d_r) \bowtie S = R \bowtie S - d_r \bowtie S$
- similar techniques for selection, projection (must maintain multiplicity counters though) and aggregation
- Blakeley et al. [SIGMOD 87], Gupta and Mumick survey [DE Bulletin 95].

Continuous Querying

- Idea: define a query, results get updated and shown to you dynamically, as base data changes
- E.g. applications:
  - network monitoring, stock monitoring
  - alerting systems (e.g., new book arrived in library)
    - better than triggers for this application
- Implementation techniques similar to materialized view maintenance
- Maier et al, SIGMOD 98 demo session

When to Use Materialized Views

- Let V = R $\bowtie$ S be materialized
- Query may V, but may still be better to replace by view definition. Eg selection on V
- Query may use R $\bowtie$ S, but may be better to replace by V
- Job of query optimizer
  - Chaudhuri et al [ICDE95]
  - Falls out as special case of multiquery optimization algos of RSSB99

Deciding What to Materialize

- maintenance cost and query cost
  - workload:
    - queries and update transactions
    - weights for each component of workload
- workload cost depends on what is materialized
- Goal: find set of views that gives minimum cost if materialized, subject to space constraints
- Note: materializing views can reduce even update costs
  - indices, and SQL assertions

Deciding What to Materialize

- History
  - Roussopolous [1982]: exhaustive A* algorithm
  - Ross, Srivastava and Sudarshan [SIGMOD 96] suggest materializing views can reduce update costs, give heuristics
  - Labio et al. [1997], Gupta [1997], Sellis et al [1997], Yang, Karlapalem and Li [1997] give various exhaustive/heuristic/greedy algorithms
  - Chaudhuri and Narsayya [1998] considers only indices, being introduced in SQL server
  - Exhaustive algos are all doubly exponential!

Caching of Query Results

- Store results of earlier queries
- Motivation
  - speed up access to remote data
    - also reduce monetary costs if charge for access
  - interactive querying often results in related queries
    - results of one query can speed up processing of another
  - caching can be at client side, in middleware, and even in a database server itself

Query Caching (Cont)

- Differences from page/object caching
  - results that are cached are defined by a (possibly complex) query
  - cost of computing different results is different --- cost of fetching a page is same for all pages
  - sizes of different results is different --- page size is fixed
- One heuristic: benefit = (recomp-cost * freq-access) / size
  - Update frequence must also be taken into account

Query Caching (Cont)

- Differences from selection of views to materialize
  - what to cache decided based on recent queries
    - => set of cached results changes dynamically
    - adapts as users change their behaviour
  - cached data may not be maintained up-to-date
    - => if base data has been updated, query optimizer must choose between recomputing cached results and incrementally computing changes

Query Caching (Cont)

- Predicate caching (Wiederhold et al 1996) and Semantic caching (Dar et al, 1996)
  - not tied to query optimizer
- ADMS (Roussopolous, 1994)
  - handles SPJ queries, with specific graph structure
- WATCHMAN (Scheurmann et al, VLDB96)
  - makes caching decisions based on cost, frequency of usage and size
  - reuses cached results only if exactly same query repeats

Query Caching (Cont)

- Dynamat (Roussopolous et al, SIGMOD 99)
  - considers caching of data cube queries
  - not general purpose unlike ADMS, but handles update costs better
- Web caching is somewhat similar
  - cached pages differ in size, and in access cost (e.g., local pages can be accessed faster)

Data Warehouses

- Characteristics:
  - Very large
  - typical schema: very large fact table, small dimension tables
  - typical query: aggregate on join of fact table and dimension tables
- Can exploit above characteristics for optimizing queries
  - e.g., join dimension tables (even if cross product), build in memory index, scan fact table, probe index. Summarize if required and output

Data Warehouses (Cont)

- Synchronized scans
  - multiple queries can share a scan of fact table
    - slow some queries down so others catch up
- Bit map indices
  - for selections on low cardinality attributes
  - e.g.: M 10011100011001
    F 01100011100110
  - idea: and-ing of bit maps is very efficient, use on bitmaps to filter to relevant tuples, retrieve them
  - Quass and O'Neill [Sigmod 1997], various DB products (DB2, Informix, ...)

Virtual Warehouses/Databases

- Data sources are numerous and distributed
  - may be accessible only via html
    - => wrappers needed
    - Stanform TSIMMIS project, Junglee, and others have built wrappers.
  - may support only limited number of access types through forms interfaces
  - site descriptions: describe what data is contained at a site Levy et al [1995].
    - Query sent only to relevant sites.

Virtual Warehouses and Databases (Cont)

- Provide user with view of a single database, which can be queried
- Underlying system must find best/good way of evaluating query

Parallel Databases

- Search space is extremely large in general
  - How to partition data
  - How to partition operations
- Two basic approaches
  - Each operation is parallelized across all nodes
  - Get best sequential plan, then parallelize
    - scheduling issues
    - pipelining issues

New Applications

- Querying semistructured data
  - XML
  - Querying on the web
    - WebSQL, WebOQL, .. (Mendelzon.., Shmueli.., Laks..)
  - Formal query languages for semi-structureed data
    - Buneman et al

Conclusions

- Query optimization has come a long way in the last 5/6 years
- Still an area of active research
  - lots of work on selection of materialized views, and caching late
  - Driving forces: Object relational DBS, Web, increasingly complex DSS queries, Data mining
  - query optimizers are still very expensive in space and time. Better approximation algorithms could help a lot.