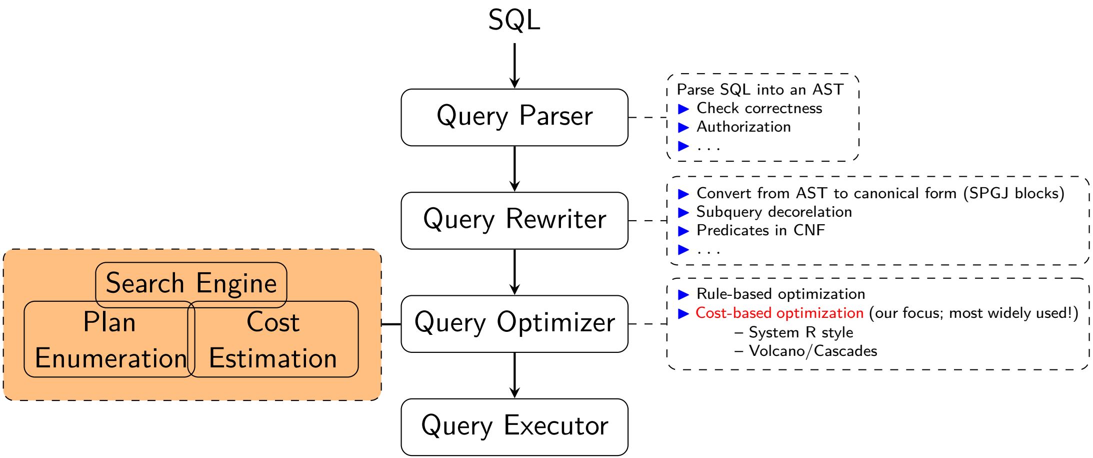
COL362/632 Introduction to Database Management Systems

Query Optimization – Cost-based Optimization

Kaustubh Beedkar

Department of Computer Science and Engineering
Indian Institute of Technology Delhi

Query Parsing & Optimization



Cost-based Optimization

Core Idea

- Enumerate <u>all</u> plans
- Estimate cost of each plan
- Pick plan with least estimated cost

Outline

Seliger Optimizer

- selection pushdown
- projections up
- Optimize join orders
  - Build plans bottom up
  - Only consider left-deep trees
    - works well with existing operator implemenations
    - Allows output of each operator to be pipelined into next operator
- Avoid cross products
- Use dynamic programming

Dynamic Programming

1. Optimality principle must hold
   - bestPlan($R \bowtie S \bowtie T$)
     $=$ bestOf((bestPlan($R \bowtie S$) $\bowtie T$), (bestPlan($R \bowtie T$) $\bowtie S$), (bestPlan($S \bowtie T$) $\bowtie R$))
2. Sub-problems must overlap

Dynamic Programming

$\mathcal{R} \leftarrow \{ R_1, \ldots, R_n \}$
for each $R \in \mathcal{R}$ **do**
  bestPlan($\{ R \}$) $\leftarrow$ AccessPaths(R)
  prune(bestPlan($\{ R \}$), costs())
for $i = 2$ to n **do**
  for $\mathcal{S} \subset \{ R_1, \ldots, R_n \}$ such that $|\mathcal{S}| = i$ **do**
    bestPlan($\mathcal{S}$) $\leftarrow \emptyset$
    for each $R \in \mathcal{S}$ **do**
      bestPlan($\mathcal{S}$) $\bigcup =$ bestPlan($\mathcal{S} \setminus R$) $\bigoplus$ bestPlan($\{ R \}$)
      prune(bestPlan($\mathcal{S}$), costs())
prune(bestPlan($\mathcal{R}$), costs())
return bestPlan($\mathcal{R}$)

Dynamic Programming (Example)

DP Table				
subset				best plans
R	S	T	U	
X				scan(R), iseek(r, R), isam(a,A)
	X			scan(S), isam(a,S)
		X		scan(T), isam (c,T)
			X	scan(U), iseek(u,U), isam(c,U)

```
select R.w,S.x,T.y,U.z
from R, S, T, U, V
where
   R.a = S.a and
   S.b = T.b and
   T.c = U.c and
   R.r = 10 and U.u > 25
```

- unclustered index on $R.r$
- unclustered index on $U.u$
- clustered index on keys (a,b,c)

Dynamic Programming (Example)

DP Table				
subset				**best plans**
R	S	T	U	
X				scan(R), iseek(r, R), isam(a,A)
	X			scan(S), isam(a,S)
		X		scan(T), isam (c,T)
			X	scan(U), iseek(u,U), isam(c,U)

```
select R.w,S.x,T.y,U.z
from R, S, T, U, V
where
   R.a = S.a and
   S.b = T.b and
   T.c = U.c and
   R.r = 10 and U.u > 25
```

- unclustered index on $R.r$
- unclustered index on $U.u$
- clustered index on keys (a,b,c)

Dynamic Programming (Example)

DP Table				
subset				**best plans**
R	S	T	U	
X				iseek(r, R)
	X			isam(a,S)
		X		isam (c,T)
			X	scan(U)
X	X			iseek(r,R)$\bowtie^{SHJ}$isam(a,S), iseek(r,R)$\bowtie^{BNLJ}$isam(a,S),. . .
X		X		iseek(r,R)$\times$isam(a,S), isam(a,S)$\times$iseek(r,R)
X			X	iseek(r,R)$\times$scan(U), scan(U)$\times$iseek(r,R)
	X	X		isam(a,S)$\bowtie^{SHJ}$isam(c,T), isam(c,T)$\bowtie^{SHJ}$isam(a,S),...
	X		X	isam(S)$\times$scan(U),...
		X	X	isam(c,T)$\bowtie^{SHJ}$scan(U), scan(U)$\bowtie^{SHJ}$isam(c,T),...

```
select R.w,S.x,T.y,U.z
from R, S, T, U, V
where
   R.a = S.a and
   S.b = T.b and
   T.c = U.c and
   R.r = 10 and U.u > 25
```

- unclustered index on $R.r$
- unclustered index on $U.u$
- clustered index on keys (a,b,c)

Dynamic Programming (Example)

DP Table				
subset				**best plans**
R	S	T	U	
X				iseek(r, R)
	X			isam(a,S)
		X		isam (c,T)
			X	scan(U)
X	X			iseek(r,R)$\bowtie^{SHJ}$isam(a,S), iseek(r,R)$\bowtie^{BNLJ}$isam(a,S),. . .
X		X		iseek(r,R) $\times$ isam(a,S), isam(a,S) $\times$ iseek(r,R)
X			X	iseek(r,R) $\times$ scan(U), scan(U) $\times$ iseek(r,R)
	X	X		isam(a,S)$\bowtie^{SHJ}$isam(c,T), isam(c,T)$\bowtie^{SHJ}$isam(a,S),...
	X		X	isam(S) $\times$ scan(U),...
		X	X	isam(c,T)$\bowtie^{SHJ}$scan(U), scan(U)$\bowtie^{SHJ}$isam(c,T),...

```
select R.w,S.x,T.y,U.z
from R, S, T, U, V
where
   R.a = S.a and
   S.b = T.b and
   T.c = U.c and
   R.r = 10 and U.u > 25
```

- unclustered index on $R.r$
- unclustered index on $U.u$
- clustered index on keys (a,b,c)

Dynamic Programming (Example)

DP Table				
subset				**best plans**
R	S	T	U	
X				iseek(r, R)
	X			isam(a,S)
		X		isam (c,T)
			X	scan(U)
X	X			iseek(r,R)$\bowtie^{SHJ}$isam(a,S)
X		X		iseek(r,R)$\times$isam(a,S)
X			X	iseek(r,R)$\times$scan(U)
	X	X		isam(c,T)$\bowtie^{SHJ}$isam(a,S)
	X		X	isam(S)$\times$scan(U)
		X	X	isam(c,T)$\bowtie^{SHJ}$scan(U)
X	X	X		(iseek(r,R)$\bowtie^{SHJ}$isam(a,S)) $\bowtie^{SHJ}$ isam(c,T)
X	X		X	(iseek(r,R)$\bowtie^{SHJ}$isam(a,S)) $\times$ scan(U)
X		X	X	(isam(c,T)$\bowtie^{SHJ}$scan(U)) $\times$ iseek(r,R)
	X	X	X	(isam(c,T)$\bowtie^{SHJ}$isam(a,S)) $\bowtie^{SHJ}$ scan(U)

```
select R.w,S.x,T.y,U.z
from R, S, T, U, V
where
   R.a = S.a and
   S.b = T.b and
   T.c = U.c and
   R.r = 10 and U.u > 25
```

- unclustered index on $R.r$
- unclustered index on $U.u$
- clustered index on keys (a,b,c)

Dynamic Programming (Example)

DP Table				
subset				**best plans**
R	S	T	U	
X				iseek(r, R)
	X			isam(a,S)
		X		isam (c,T)
			X	scan(U)
X	X			iseek(r,R)$\bowtie^{SHJ}$isam(a,S)
X		X		iseek(r,R)$\times$isam(a,S)
X			X	iseek(r,R)$\times$scan(U)
	X	X		isam(c,T)$\bowtie^{SHJ}$isam(a,S)
	X		X	isam(S)$\times$scan(U)
		X	X	isam(c,T)$\bowtie^{SHJ}$scan(U)
X	X	X		(iseek(r,R)$\bowtie^{SHJ}$isam(a,S)) $\bowtie^{SHJ}$ isam(c,T)
X	X		X	(iseek(r,R)$\bowtie^{SHJ}$isam(a,S)) $\times$ scan(U)
X		X	X	(isam(c,T)$\bowtie^{SHJ}$scan(U)) $\times$ iseek(r,R)
	X	X	X	(isam(c,T)$\bowtie^{SHJ}$isam(a,S)) $\bowtie^{SHJ}$ scan(U)
X	X	X	X	

```
select R.w,S.x,T.y,U.z
from R, S, T, U, V
where
   R.a = S.a and
   S.b = T.b and
   T.c = U.c and
   R.r = 10 and U.u > 25
```

- unclustered index on $R.r$
- unclustered index on $U.u$
- clustered index on keys (a,b,c)

Interesting Orders

Interesting orders

- Physical property of a relation, e.g., Relation sorted on some attribute
- Breaks the principle of optimality
- Intermediate relation has an interesting order, if the order can be used later to
  - sort later (`order by`)
  - `group by`
  - perform merge join

Dynamic programming w/ Interesting Orders

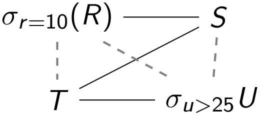
- For each subset of relations, compute multiple optimal plans (one for each interesting order)

Exploiting Query Graph Structure

Query Graph

- Undirected graph with $R_1, \ldots, R_n$ as nodes
- Join predicate of the form $R_i.a = Rj.b$ forms an edge between R_i and R_j
- Predicate of the form $R_i.a = c$ are pushed down

Example



Outline

Design Principles

Focus on extensibility

1. Query processing grounded in algebraic techniques
   - Define new algebra operators, equivalence rules, operator implementation algorithms
2. Rules
3. Based on algebraic equivalences
4. Parameterized rule compilation
5. Dynamic programming based search engine (top-down approach)

Concepts

Expressions

- Some query operation with zero or more input expression

  e.g., logical expression $R \bowtie S$

  e.g., physical expression iseek$(id, R) \bowtie^{SHJ}$ isam(id, S)

Rules

1. Trasformation Rules
   - E.g, $R \bowtie S \rightarrow S \bowtie R$
2. Implementation Rules
   - E.g., $R \bowtie S \rightarrow R \bowtie^{SMJ} S$

- Each rule is specified as
  - pattern that defines the structure of the expression
  - and resulting transformtion

Concepts

Properties

- Logical properties
  - derived from logical algebra expression
- Physical properties
  - depend on algorithms, e.g, sort order

Enforcers

- Ensure property, e.g., sort, partitioning
- can also destroy properties

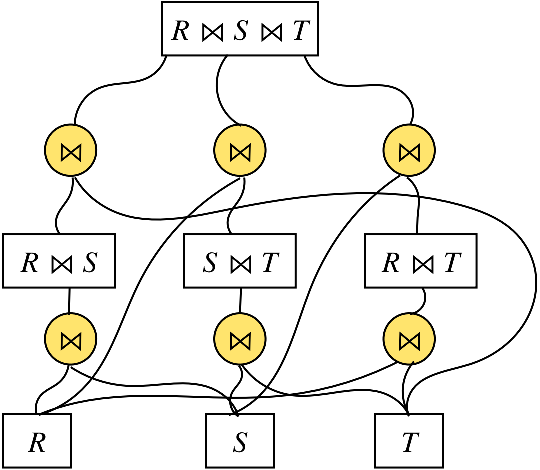
AND-OR DAG

Directed Acyclic Graph (DAG)

- Equivalance nodes (OR-nodes)

  Operation nodes (AND-nodes) as children
- Operation nodes

  Equivalence nodes as children



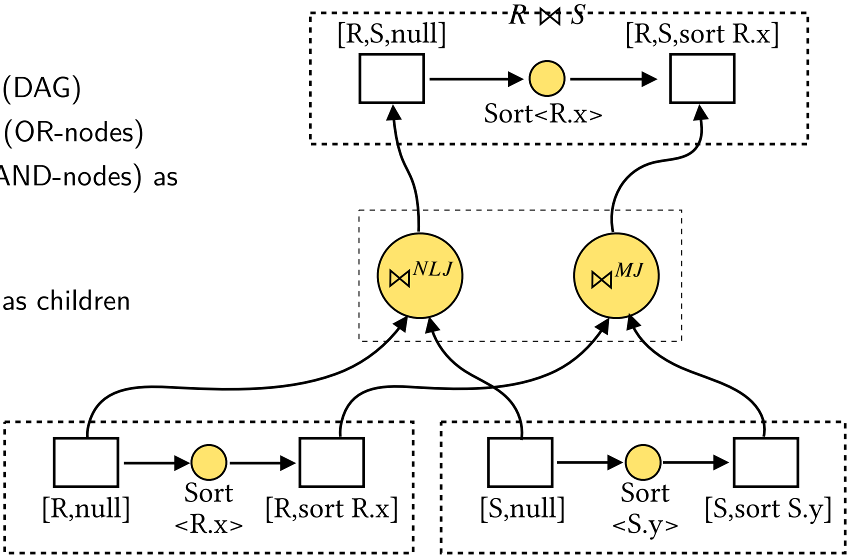
AND-OR DAG

Directed Acyclic Graph (DAG)

- Equivalance nodes (OR-nodes)

  Operation nodes (AND-nodes) as children
- Operation nodes

  Equivalence nodes as children



Finding the Best Plan

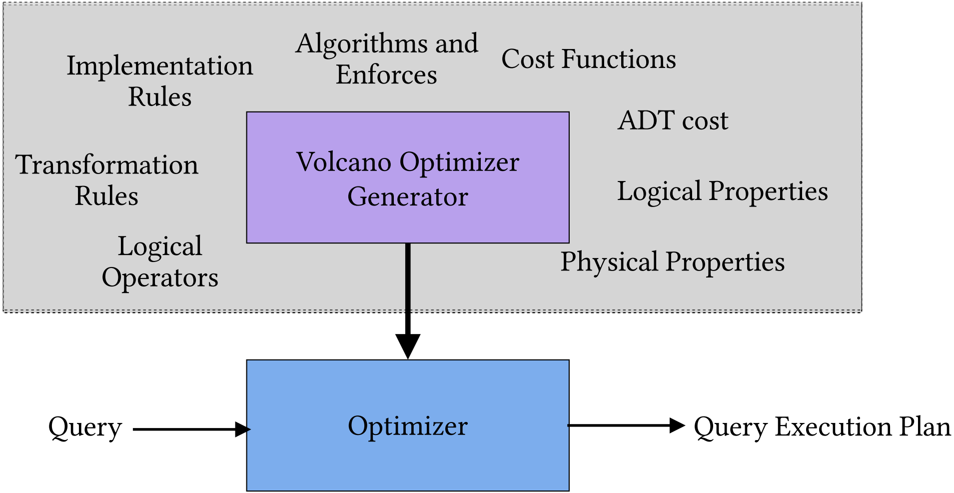
- DAG generation interleaved with finding the best plan

  Directed dynamic programming

  Branch and bound pruning

More than an Optimizer

Optimizer generator framework



Outline

Nested Subqueries

Example nested subquery

```
select name from supplier s
where exists (
   select * from partsupplier ps
   where s.sid = ps.sid and ps.qty > 100
  )
```

- Attribute from an outer relation is used in the inner subquery
- Correlated evaluation: evaluate the outer query and invoke the inner query — can be very inefficient! Why?

Nested Subqueries

Example nested subquery

```
select name from supplier s
where exists (
   select * from partsupplier ps
   where s.sid = ps.sid and ps.qty > 100
  )
```

- Attribute from an outer relation is used in the inner subquery
- Correlated evaluation: evaluate the outer query and invoke the inner query — can be very inefficient! Why?

Subquery de-correlation

- Turn nested query into a join, this is not always possible!

Nested Subqueries

Example nested subquery

```
select name from supplier s
where exists (
   select * from partsupplier ps
   where s.sid = ps.sid and ps.qty > 100
  )
```

- Attribute from an outer relation is used in the inner subquery
- Correlated evaluation: evaluate the outer query and invoke the inner query — can be very inefficient! Why?

Subquery de-correlation

- Turn nested query into a join, this is not always possible!
- Use semi-joins instead! (Recall semantics of semi-join operator)

Materialized Views

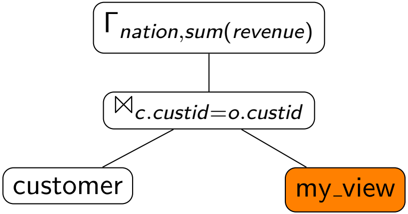
- Which views to materialize?
  – similar problem: which indexes to create?

- Optimizing queries using materialized views

Example

```
create materialized view my_view as
select o.custid, count(*), sum(l.qty*o.price)
from lineitem l, orders o
where l.oid=o.oid group by o.custid

select c.nation, sum(revenue) from customer c,
   (select o.custid, sum(l.qty*o.price) as revenue
   from lineitem l, orders o where l.oid = o.oid
   group by o.oid) as iq
where c.custid = iq.custid group by c.nation
```



Outline

Postgres Query Plans (Self Study)

- Using explain statements https://www.postgresql.org/docs/current/sql-explain.html
- Explore plan Visualizer https://explain.dalibo.com/
- Creating Indexes https://www.postgresql.org/docs/current/sql-createindex.html
- Creating statistics https://www.postgresql.org/docs/current/sql-createstatistics.html