Discussion 7

Query Optimization

Jiahui Xu

Query Optimization

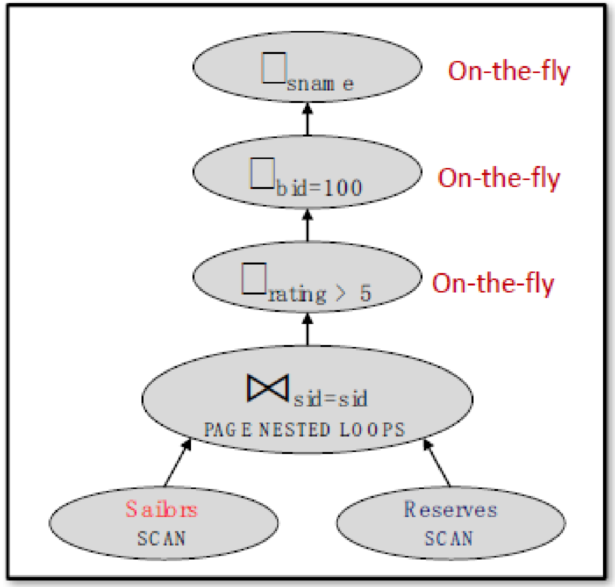
- **Three beautifully orthogonal concerns:**
  - Plan space:
    - for a given query, what plans are considered?
  - Cost estimation:
    - how is the cost of a plan estimated?
  - Search strategy:
    - how do we “search” in the “plan space”?
- **Optimization goal:**
  - Ideally: Find the plan with least actual cost.
  - Reality: Find the plan with least estimated cost.
    - And try to avoid really bad actual plans!

Background

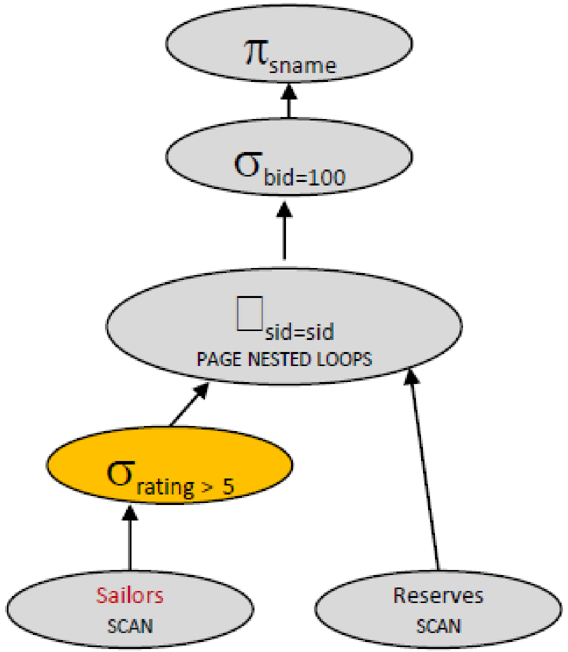
- Query block can be converted to relational algebra
- Rel. Algebra converts to tree
- Operators can also be applied in different orders!
- Order of operators affects I/Os and resource usage, but not necessarily output

Plan Space

- Selection/ Projection cascade and pushdown
  - The earlier we reduce the size of our input data, the fewer I/Os are incurred as we traverse up the tree
  - Pushing a selection into the inner loop of a nested loop join doesn't save I/Os! Essentially equivalent to having the selection above.
- Avoid Cartesian products
  - Given a choice, do theta-joins rather than cross-products



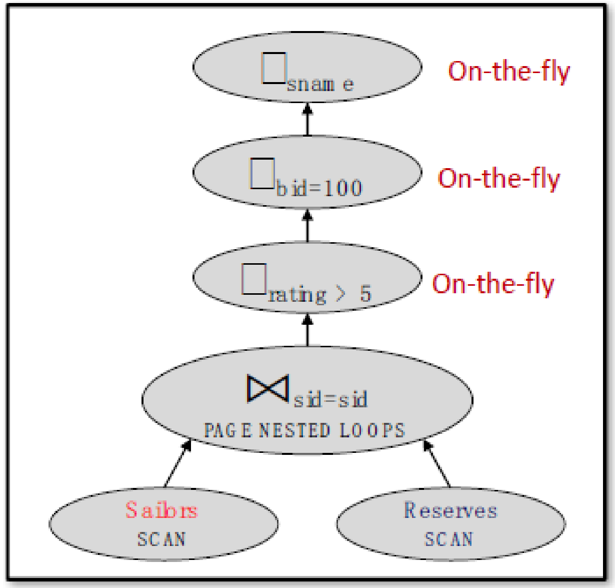
500,500 IOs



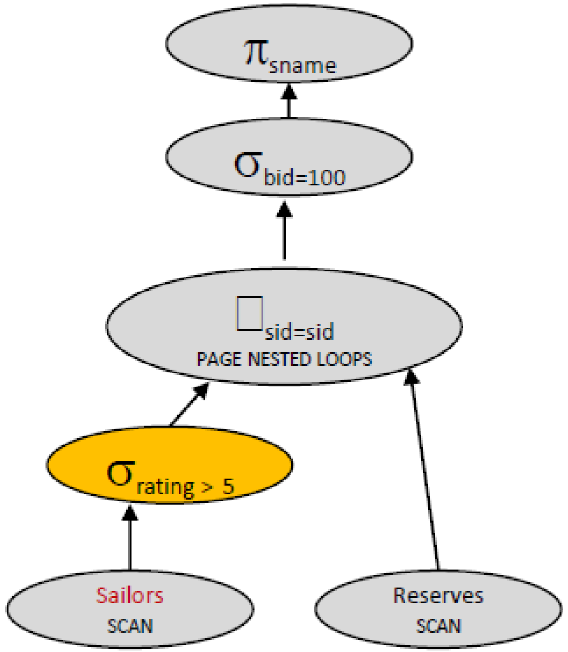
250,500 IOs

Plan Space – What can we do?

- Selection/ Projection cascade and pushdown
  - **The earlier we reduce the size of our input data, the fewer I/Os are incurred as we traverse up the tree**
  - Pushing a selection into the inner loop of a nested loop join doesn't save I/Os! Essentially equivalent to having the selection above.



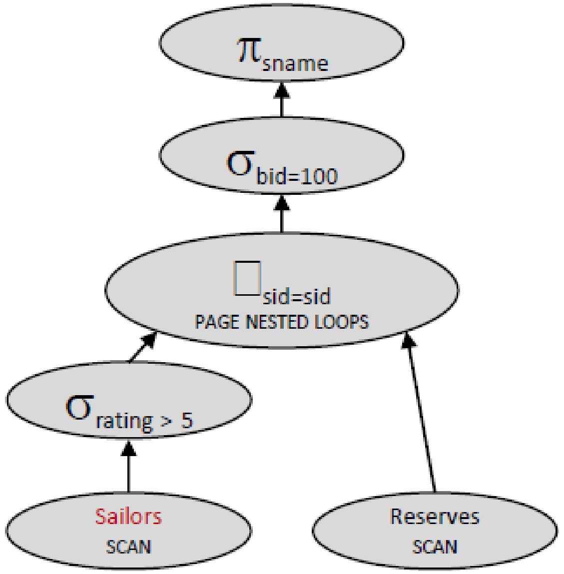
500,500 IOs



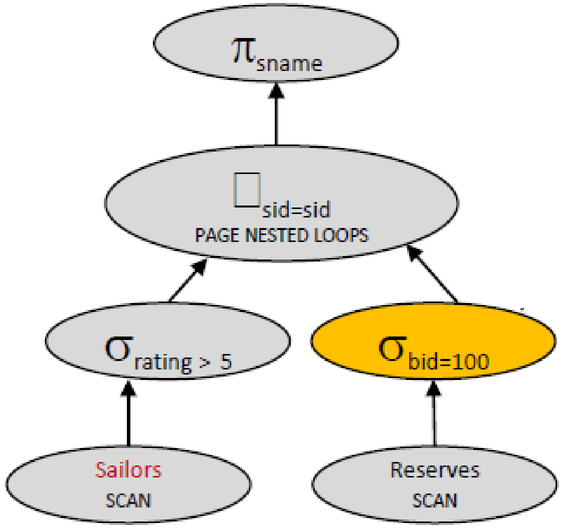
250,500 IOs

Plan Space – What can we do?

- Selection/ Projection cascade and pushdown
  - The earlier we reduce the size of our input data, the fewer I/Os are incurred as we traverse up the tree
  - **Pushing a selection into the inner loop of a nested loop join doesn’t save I/Os! Essentially equivalent to having the selection above.**



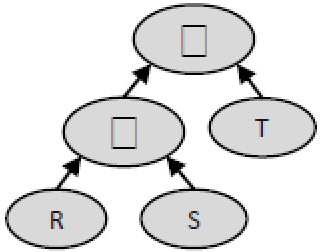
250,500 IOs

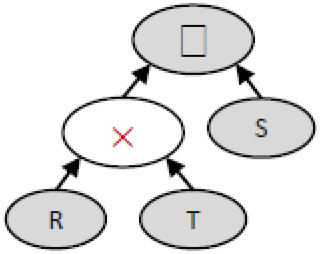


250,500 IOs

Plan Space – What can we do?

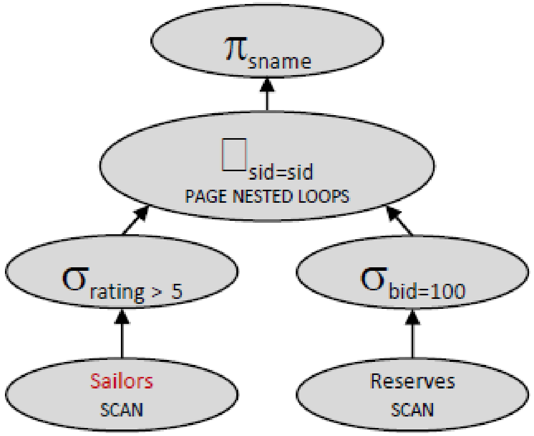
- Avoid Cartesian products
  - Given a choice, do theta-joins rather than cross-products

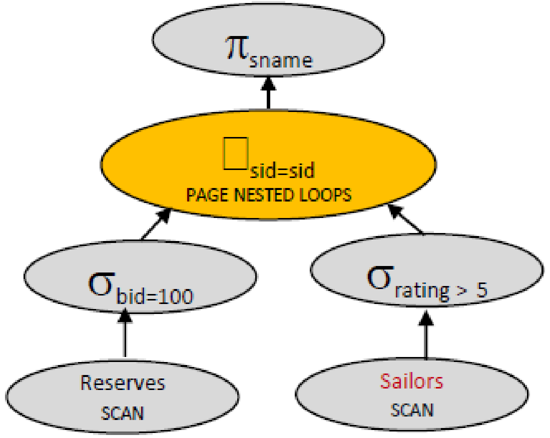




Plan Space – What can we do?

- Exchange Join Order
- Scan A (Npage(A) IOs)
- For each pageful of A for conditionA, Scan B (Npage(B) IOs)
- Total: Npage(A) + Npage(A) * selectivity(conditionA) * Npage(B))



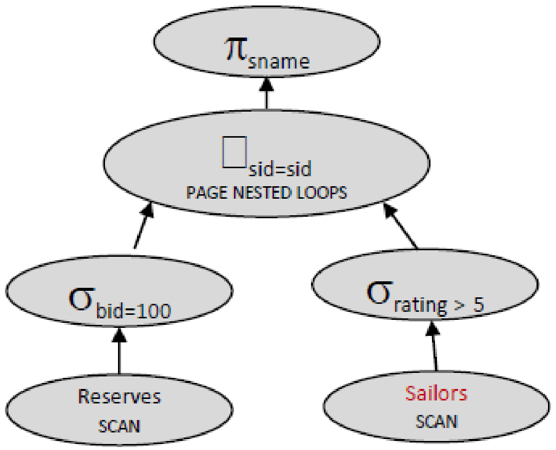


500 + 250*1000= 250,500 IOs

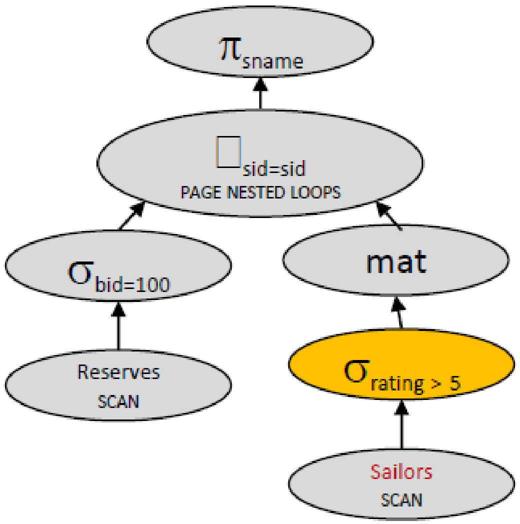
6000 IOs =1000 +10*500

Plan Space – What can we do?

- Materializing Inner Loops(write to a temp table)



6000 IOs



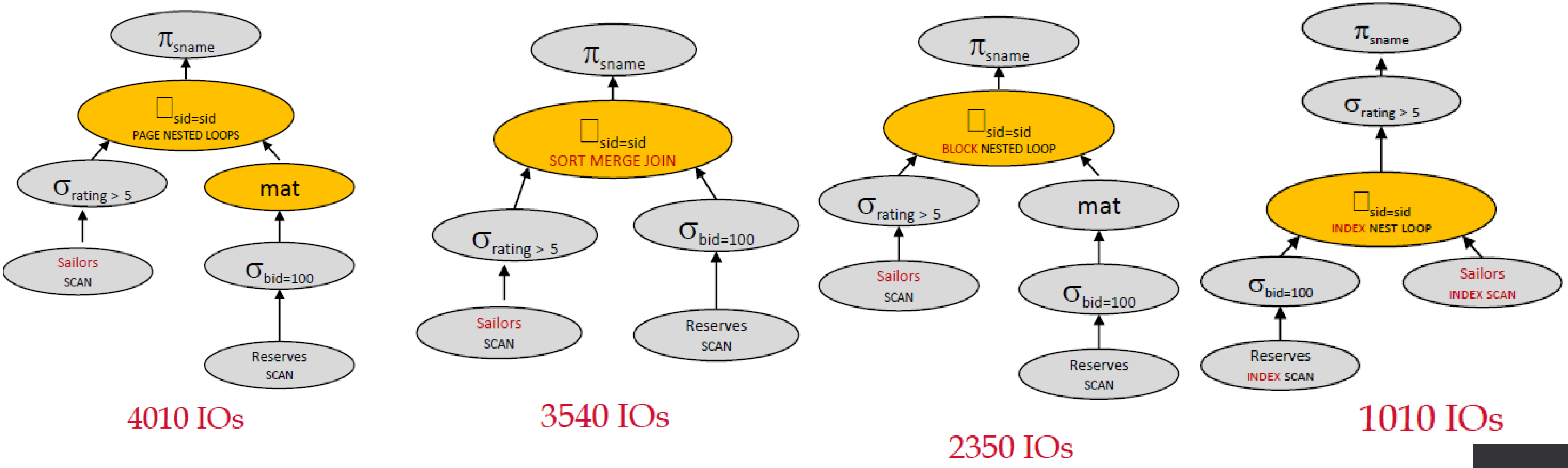
4250 IOs

- Scan Reserves (1000 IOs)
- For each pageful of Reserves for bid 100,
- Scan Sailors (500 IOs)
- 1000 +10*500

- Scan Reserves (1000 IOs)
- Scan Sailors (500 IOs)
- Materialize Temp table T1 (250 IOs)
- For each pageful of Reserves for bid 100(10), Scan T1 (250 IOs)
- 1000 + 500+ 250 + (10 * 250)

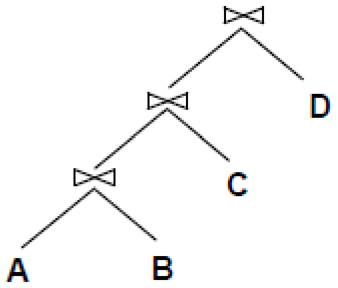
Plan Space – What can we do?

- Join Algorithm
  - Use indices (e.g. INLJ)



System R optimizer

- We’ll be looking at the System R optimizer
- Plan space:
  - only left-deep trees, avoid cartesian products unless they’re the only option.
  - Left-deep trees represent a plan where all new tables are joined one at a time from the right.



- Cost estimation:
  - actual Selinger optimizer incorporates both CPU and I/O cost
  - we’ll only use I/O cost for this class
- Search algorithm: dynamic programming

Cost Estimation

- For each plan considered, must estimate total cost:
- Must estimate ***cost*** of each operation in plan tree: Depends on input cardinalities.
  - We've already discussed this for various operators: sequential scan, index scan, joins, etc.
- Must estimate ***size of result*** for each operation in tree: it determines downstream input cardinalities
  - Use information about the input relations.
  - For selections and joins, assume independence of predicates.

Cost Estimation

- Max output cardinality = product of input cardinalities
- **Selectivity (sel)** associated with each **term**
  - reflects the impact of the term in reducing result size.
  - selectivity = |output| / |input|
  - Book calls selectivity “Reduction Factor”(RF)
  - Assume: **uniform** distribution within histogram bins
- **Result Cardinality**
- = # of output tuples
- = Max # tuples * product of all selectivitie s

Selectivity

11.1 Equalities

Predicate	Selectivity	Assumption
$c = v$	1 / (# of distinct values of c in index)	We know $\lvert c \rvert$.
$c = v$	$1/10$	We don't know $\lvert c \rvert$.
$c_1 = c_2$	1 / max(# of distinct values of c_1, # of distinct values of c_2)	We know $\lvert c_1 \rvert$ and $\lvert c_2 \rvert$.
$c_1 = c_2$	1 / (# of distinct values of c_i)	We know $\lvert c_i \rvert$ but not $\lvert$other column$\rvert$.
$c_1 = c_2$	$1/10$	We don't know $\lvert c_1 \rvert$ or $\lvert c_2 \rvert$.

11.2 Inequalities on Integers

Predicate	Selectivity	Assumption
$c < v$ $c > v$	(v - min(c)) / (max(c) - min(c) + 1) (max(c) - v) / (max(c) - min(c) + 1)	We know max(c) and min(c). c is an integer.
$c < v$ $c > v$	$1/10$	We don't know max(c) and min(c). c is an integer.
$c <= v$ $c >= v$	(v - min(c)) / (max(c) - min(c) + 1) + (1/$\lvert c \rvert$) (max(c) - v) / (max(c) - min(c) + 1) + (1/$\lvert c \rvert$)	We know max(c) and min(c). c is an integer.
$c <= v$ $c >= v$	$1/10$	We don't know max(c) and min(c). c is an integer.

Selectivity

11.3 Inequalities on Floats

Predicate	Selectivity	Assumption
$c >= v$	(max(c) - v) / (max(c) - min(c))	We know max(c) and min(c). c is a float.
$c >= v$	1/10	We don't know max(c) and min(c). c is a float.
$c <= v$	(v - min(c)) / (max(c) - min(c))	We know max(c) and min(c). c is a float.
$c <= v$	1/10	We don't know max(c) and min(c). c is a float.

11.4 Connectives

Predicate	Selectivity	Assumption
p1 **AND** p2	S(p1)*S(p2)	Independent predicates
p1 **OR** p2	S(p1) + S(p2) - S(p1 AND p2)	
NOT p	1 - S(p)	

Cost Estimates for Single-Relation Plans

- Index I on primary key matches selection:
  - Cost is (Height(I) + 1) + 1 for a B+ tree.
- Clustered index I matching selection:
  - (NPages(I)+**NPages**(R)) * selectivity.
- Non-clustered index I matching selection:
  - (NPages(I)+**NTuples**(R)) * selectivity.
- Sequential scan of file:
  - NPages(R).
- Recall: Must also charge for duplicate elimination if required

Search algorithm: dynamic programming (bottom up)

- Left deep & No cartesian products
- Enumerated using N passes (if N relations joined):
  - **Pass 1:** Find best 1-relation plan for each relation
  - **Pass i:** Find best way to join result of an (i-1)-relation plan (as outer) to the i' threlation. (i between 2 and N.)
- For each subset of relations, retain only:
  - Cheapest plan overall, plus
  - Cheapest plan for each *interesting order* of the tuples.
    - ORDER BY attributes
    - GROUP BY attributes
    - Join attributes of yet to be added joins
      - subsequent merge join might be good