Lec 15 Query Planning and Execution

Example

Consider
```
select distinct ename
from Emp E, Dept D
where E.did = D.did and D.dname = "Toy"
```

And catalogue with indexing info

clustered → ▲ △ △ ← nonclustered

Emp (ssn, ename, ..., did) 10k records 1k pages

Dept (▲did, △dname, ...) 500 records 50 pages

▷ Naïve ~ 2M IO

```
↑ πename                 — 4 reads  1 write
↑ σdname = "Toy"         — 2k reads, 4 writes   (200 records if uniform)
↑ σE.did = D.did         — 1M reads, 2k writes  (10k records)
→ × ←                    — 50 + 50k reads, 1M writes  (5M records)
E   D
```

▷ Ok ~ 54k IO

```
↑ πename
↑ σdname = "Toy"
↑ ⋈ E.did = D.did
→ ← 
E   D
```

4 reads 1 write
2k reads, 4 writes
50 + 50k reads, 2k write [nested loop join] △
↑ assume fit in buf pool

△ or [sort merge join] ⇒ ~ 7.2k IO ▫
▫ and if vectorisation ⇒ ~ 3.2 IO

▷ Change tree ~ 37 IO

```
↑ π                                  — 4 reads, 1 write
↑ ⋈ E.did = D.did                    — 1 + 3 + 20 reads, 4 writes
↗ ↖
σdname = "Toy"   E
↑
D
```
[use index (dname)], 3 reads, 1 write

* Annotated RA tree i.e. Physical Plan
  └ The tree + (access plan, estimated cardinality) per node

Query Optimisation

Equivalent plan space for same query is NP-hard w.r.t. num of joins.

Approaches:

- Heuristic, Rules
  → Push σ down as much as possible
  - No need to examine data
- Cost-based
  └ come up with trees, estimate, pick cheapest

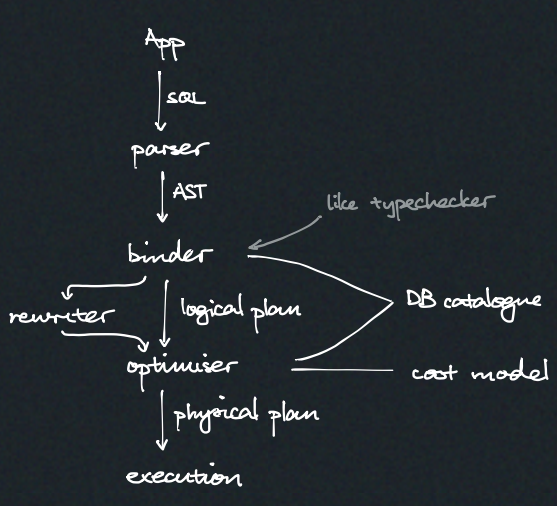
▷ Rules

→ Predicate push down
→ Replace cartesian product

$\sigma_{A.id = B.id}(A \times B) \rightarrow \bowtie_{A.id = B.id}$

→ Projection push down
⋮

Architecture

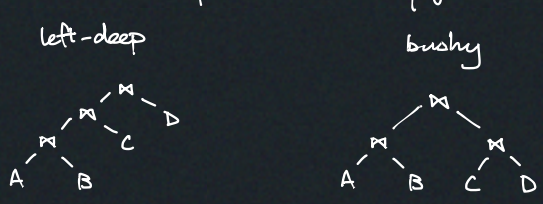


Single-rel query planning

Choose good access method

System R query optimisation (bottom-up approach)

1. Find best access path for each col on each table
2. Construct all possible left-deep join



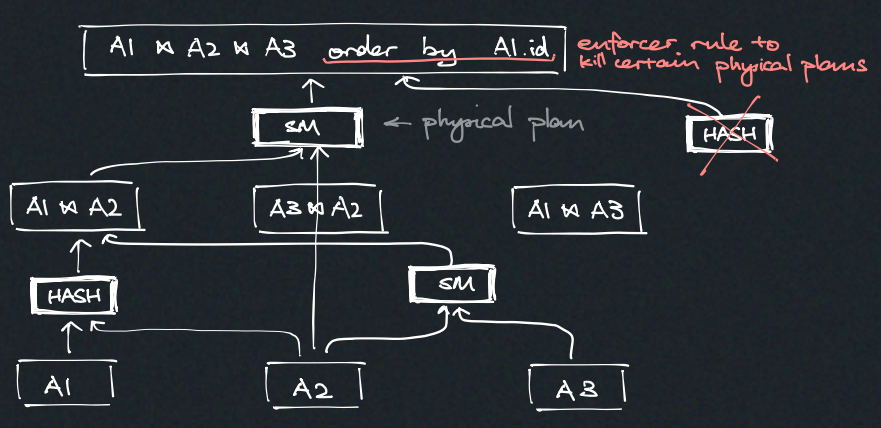
3. Find min cost among those
   └ use DP to keep best access path at each node

Multi-rel query optimisation (top-down approach)

Use rules first, then do DP

Ex. we want $\sigma_{A1 \bowtie A2 \bowtie A3 \text{ order by } A1.id}$ ← this is logical

Top-down traverse tree, apply either { logical → logical / logical → physical } rules



+ can prune without searching full space
+ can have more rules like enforcer
- harder to write

Rewriting

▷ Nested sub-queries rewrite

Do transformer to rewrite query

Usually rule-based, like elaborator

▷ Decomposing queries

Simply pull out nested query, and optimise top queries separately

▷ Unreachable handling

- 1 == 0 ⇒ false
- x > 5 && x > 10 ⇒ x > 10

Cost model

- Many magical numbers
- May take CPU costs
- Use internal statistics
- Build histogram
  └ allows estimating num records after range selection
    → equi-width hist. — same sized buckets
    → quantiles — about uniform buckets but diff. range
  └ independence assumptions ?
  └ uniform .. ?
  → HyperLogLog
- Sampling to guess count