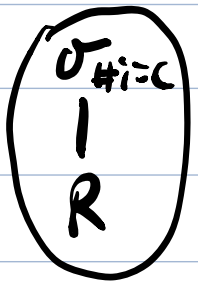
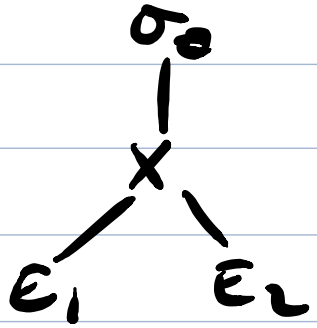
deduplication can be implemented in various ways:
- hashing
- sorting

in any case, it's expensive (need temp data struct)

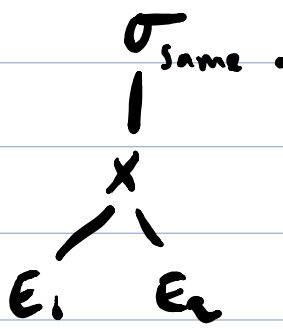


↝ B+ tree access

Additional Operators
- Deduplicate(E)
- $R(\#i = c)$
- $E_1 \bowtie_\theta E_2 \equiv$



- $E_1 \bowtie E_2$
  ↑ natural join
  $\equiv$

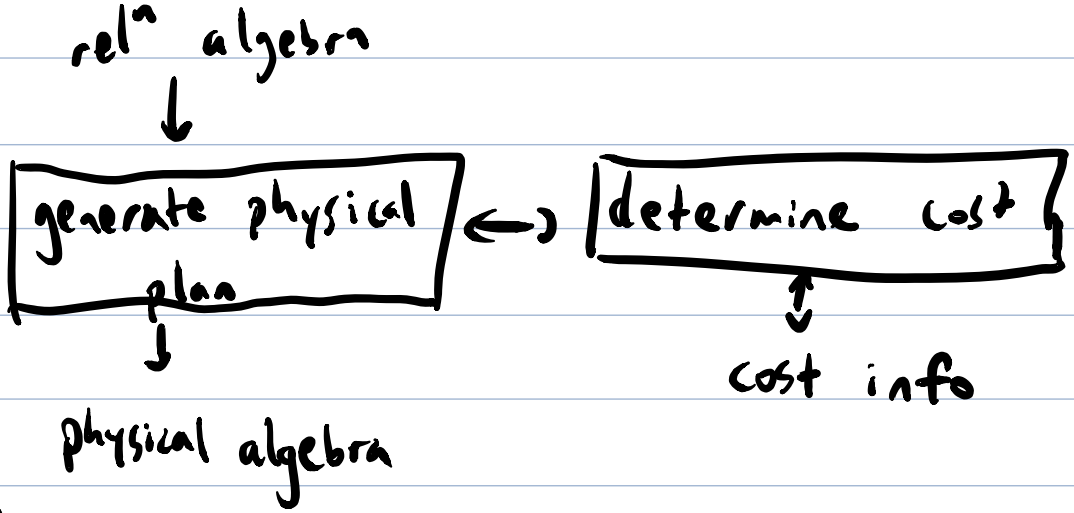


- $E_1 \div E_2$ (like universal quantification)

syntactic sugar only

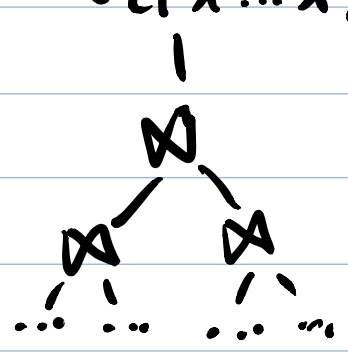
Finding an optimal query plan is computationally infeasible ⇒ we look for a <u>reasonable</u> one

- db engine has statistical metadata, helps estimate relative cost of plans



finding all equivalent plans is undecidable in general!

- expensive even for conjunctive queries
  - ↳ the <u>join-ordering problem</u>: given a selection of conditions



find best ordering of the child joins (NP-hard)

⇒ use heuristics

simple cost model for disk I/O, assume:

uniformity: all possible attrib equally likely

independence: likelihood of attrib. having a value in a tuple does not depend on values of other attribute

⇒ pretty draconian assumptions

b(R) := blocking factor for stored rel^n R
(describes # of entries in a page on disk or in B+-tree)

$\Pi_?$

σ

|

CourseInd

selecting N tuples from rel^n R using primary index has a cost of $2 + \frac{N}{b(R)}$

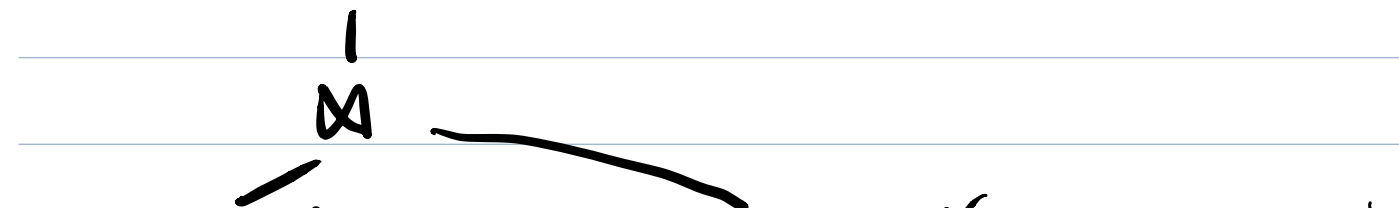
in this case $2 + \frac{100}{b(Mark)} = 2 + \frac{100}{50} = \boxed{4}$

$\frac{|Mark|}{100} = 100$ tuples for PHYS

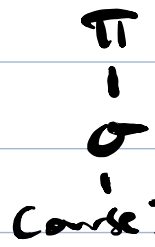
$\Pi_?$

|

$\sigma_?$



Cost = 2 + N = 21



200 block I/O operations

$$\text{cost}(\sigma_C(E)) = (1 + \varepsilon_C)\,\text{cost}(E)$$

$$\text{Cost}(R \times S) = \text{cost}(R) + \frac{|R|}{b}\,\text{cost}(S)$$

$$\approx |R|\,|S|$$

$$\text{cost}(R \bowtie S) = \text{cost}(R) + ds\,|R| \quad \text{index join}$$

Sort-merge join

$$\text{Cost}(R \bowtie S) = \text{cost}(\text{sort}(R)) + \text{cost}(\text{sort}(S))$$

where $\text{cost}(\text{sort}(E)) = \text{cost}(E) + \frac{|E|}{b} \lg \frac{|E|}{b}$

Definition (Selectivity)

$$\text{sel}(\sigma_C(R)) = \frac{|\sigma_C(R)|}{|R|}$$

Always good: push selections down

$$\sigma_\psi(E_1 \bowtie_\theta E_2) = \sigma_\psi(E_1) \bowtie_\theta \sigma_\psi(E_2)$$

push projections down
replace products by joins