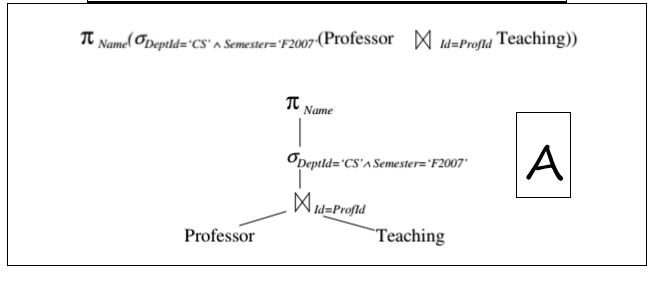
**The SQL given below is** to retrieve names of professor belong to Computer Science (CS) who are teaching students studying in semester F2007. Two different **Relational Algebra Expression** and **its Query Tree A** **and B** are shown below.

**SELECT P. Name**

**FROM Professor P, Teaching T**

**WHERE P.Id = T. ProfId -- join condition**

**AND P. DeptId = ‘CS’ AND T. Semester = ‘F2007’**



Metadata on Tables (in system catalogue)

* Professor (Id, Name, DeptId)

• size: 200 pages, 1000 rows, 50 departments

• indexes: clustered, 2-level B+tree on DeptId, hash on Id

* Teaching (ProfId, CrsCode, Semester)

• size: 1000 pages, 10,000 rows, 4 semesters

• indexes: clustered, 2-level B+tree on Semester; hash on ProfId

* Definition: Weight of an attribute – average number of rows that have a particular value

• weight of Id = 1 (it is a key)

• weight of Prof Id = 10 (10,000 classes/1000 professors)

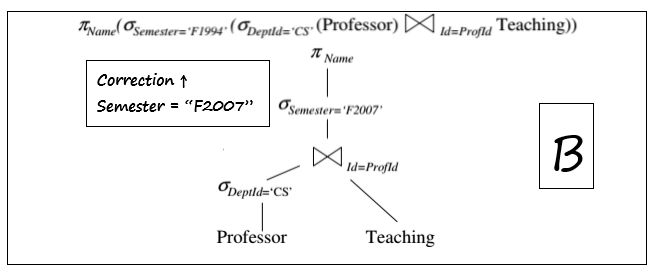
Estimating Cost - Example

* Join - block-nested loops with 51 page buffer
* Scanning Professor (outer loop): 200 page transfers, (4 iterations, 50 transfers each)
* Finding matching rows in Teaching (inner loop): 1000 page transfers for each iteration of outer loop

• 250 professors in each 50 page chunk \* 10 matching Teaching tuples per professor = 2500 tuple fetches = 2500 page transfers for Teaching (Why?)

• By sorting the record Ids of these tuples we can get away with only 1000 page transfers (Why?)

* total cost = 200+4\*1000 = 4200 page transfers



selection

Compute σ DeptId=‘CS’ (Professor) (to reduce size of one join table) using clustered, 2-level B+ tree on DeptId.

50 departments and 1000 professors; hence weight of DeptId is 20 (roughly 20 CS professors). These rows are in ~4 consecutive pages in Professor. • Cost = 4 (to get rows) + 2 (to search index) = 6 • keep resulting 4 pages in memory and pipe to next step

join

Index-nested loops join using hash index on ProfId of Teaching and looping on the selected professors (computed on previous slide) Since selection on Semester was not pushed, hash index on ProfId of Teaching can be used Note: if selection on Semester were pushed, the index on ProfId would have been lost – an advantage of not using a fully pushed query execution plan

Each professor matches ~10 Teaching rows. Since 20 CS professors, hence 200 teaching records. All index entries for a particular ProfId are in same bucket. Assume ~1.2 I/Os to get a bucket. • Cost = 1.2 × 20 (to fetch index entries for 20 CS professors) + 200 (to fetch Teaching rows, since hash index is unclustered) = 224

select/project

Pipe result of join to select (on Semester) and project (on Name) at no I/O cost Cost of output same as for Example 1 Total cost: 6 (select on Professor) + 224 (join) = 230 Comparison: 4200 (example 1) vs. 230 (example 2) !!!