B561 – Advanced Database Concepts – Fall 2011

Assignment 3

Enrique Areyan

Cardinality Estimation:

Let us consider N_s to be the total number of students in the Student relation. Then,

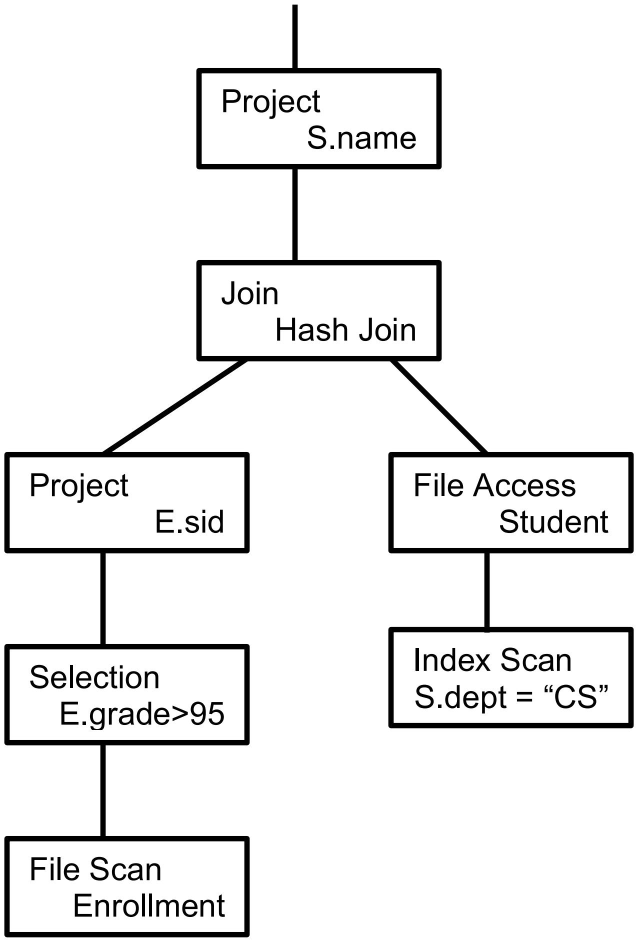
$$\frac{N_s}{50} = N_s \cdot 0.02 \cong average\ number\ of\ students\ per\ department \approx number\ of\ students\ in\ CS$$

$$N_s \cdot 0.002 \cong total\ number\ of students\ with\ grades\ greater\ than\ 90$$

$$0 \leq students\ in\ CS\ with\ grade\ greater\ than\ 90 \leq N_s \cdot 0.002$$

Thus, we are only looking to retrieve at most 0.2% of all students' records (in the case all of the students with grades above 95 are from CS). It is more likely that in the average we'll retrieve fewer than this number of records.

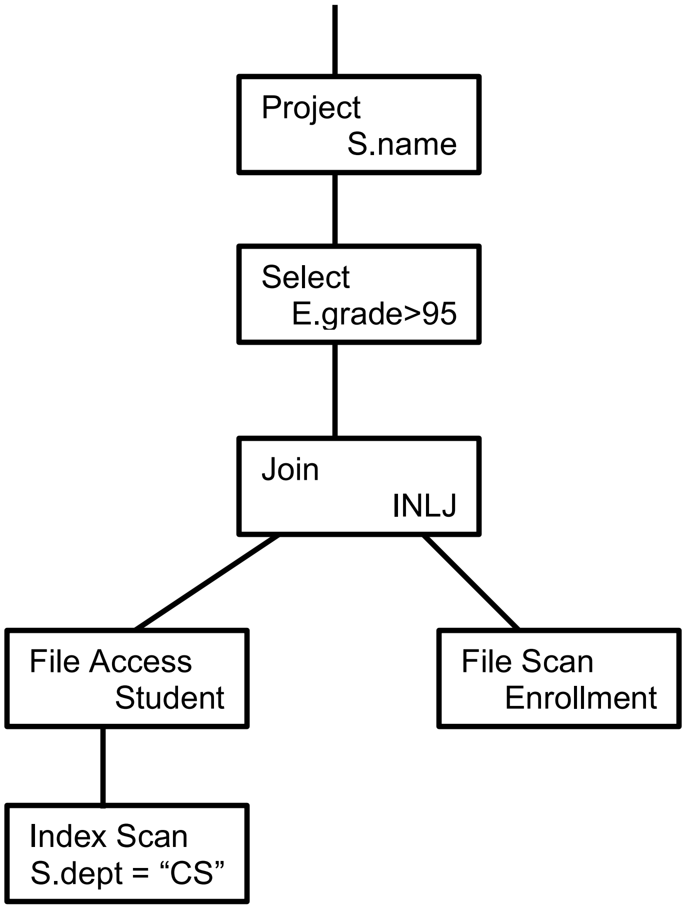
Case 1. Hash index on S.dept, Hash index on E.grade, no other index available. (20 points)



In Case 1, the hash index on E.grade is not useful because we have a range condition on grade. However, the hash index on S.dept is useful, and can help us to quickly search for students in the CS department.

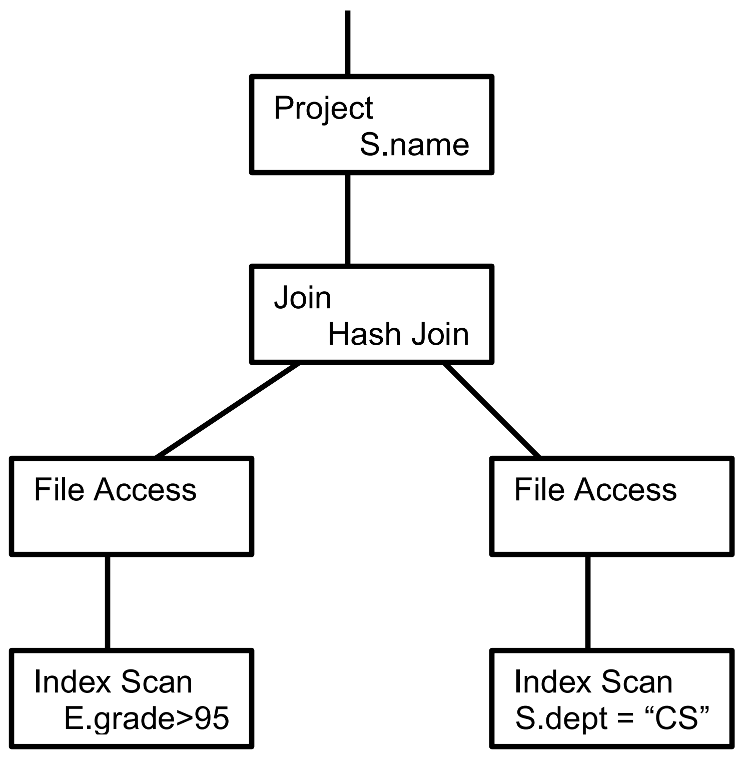
To join the results I used a Hash Join algorithm. The inner relation is the CS's Students and the outer is Enrollment such that grade is > 95. Given the cardinality estimation at the beginning, it is reasonable to assume that the CS's Students table fits into the buffer. The hash will be built on the sid of CS Students table, and for each enrollment record, the hash function will map E.sid to the appropiate bucket. If there is a match, return the tuple if not, discard it. For this strategy to be efficient, I assume that there exists a good hashing function *h*, that distributes well the sid.

Case 2. Clustered B+ tree index on E.sid, B+tree on S.dept, no other index available. (20 points)



In Case 2 both indexes are helpful but for different purposes. The index on S.dept allows us to look for the students in the CS department faster. We join the tuples with an INLJ, where the inner relation is the enrollment table that has the index on the join attribute E.sid. For each tuple in the CS student, we compare S.sid with the index on E.sid to quickly find any corresponding matches. Then we select those tuples with grades > 95.

Case 3. B+tree index on S.dept, B+tree index on E.grade, no other index available. (10 bonus points)



We have B+Tree indexes in the students and enrollment tables, both of which are useful. B+Tree works for both point and range selection conditions. I first find those tuple matching the given conditions and then join them using a Hash Join. The inner table will be the student table, for which a hash will be built on sid. We can assume that this table fits into the buffer, so the hash join is a reasonable choice. Then, for each enrollment tuple we will hash E.sid and compare directly with the bucket on the inner table.