Extra exercises 8: Query optimization

Question 1. When we say "query optimization is NP-complete", we mean that finding the optimal query plan takes exponential time with respect to the number of operations.

Question 2. When considering nested projections, cascading allows us to drop all projections but the innermost one.

Question 3. When rewriting query blocks, it is more convenient to work with them as relational algebra expressions than SQL statements.

Question 4. Table catalogues contain exact information about the table and its indexes (number of rows, min/max values in each key).

Question 5. Consider two relations R(id, rvalue) and S(id, svalue). Assume that tuples in R have rvalue that are numbers from 1 to 20, and tuples in S have svalue that are numbers from 1 to 50.
The result size estimation of the query

SELECT * FROM R, S where R.rvalue = S.svalue

is:

- A. (#tuples in R) * (#tuples in S) / 20
- B. (#tuples in R) * (#tuples in S) / 50
- C. (#tuples in R) * (#tuples in S) / (50 * 20)
- D. (#tuples in R) * (#tuples in S) / (50 + 20)

Question 6: Select the **incorrect** equivalence between relational algebra expressions.

- A.

$$\pi_{\text{T.name, S.group}}\left(\sigma_{\text{T.value } <100}\left(S \underset{\text{S.value } = \text{ T.value}}{\bowtie} T\right)\right) \equiv (\pi_{\text{name}}(\sigma_{\text{value } <100}(S))) \underset{\text{S.value } = \text{ T.value}}{\bowtie} (\pi_{\text{group}}(S))$$

- B.

$$\sigma_{\text{T.value } < \text{ S.value}}(\pi_{\text{T.value, T.name, S.value, S.name}}(S \times T)) \equiv (\pi_{\text{value, name}}(S)) \underset{\text{S.value } > \text{ T.value}}{\bowtie} (\pi_{\text{value, name}}(T))$$

- C.

$$\pi_{\text{S.name, T.name}}\left(\sigma_{\text{S.id } \leq 100}\left((\sigma_{\text{value } <100}(S)) \underset{\text{S.value } = \text{ T.value}}{\bowtie} T\right)\right) \equiv \pi_{\text{S.name, T.name}}\left((\sigma_{\text{id } \leq 100 \, \wedge \, \text{value } <100}(S)) \underset{\text{S.value } = \text{ T.value}}{\bowtie} T\right)$$

- D. All above equalities are correct.

Question 7:

Consider the following relation

Hotels(id, name, price)

Hotels have 1000 pages, 50.000 tuples. Assume that price is uniformly distributed from 1 to 200.

Consider the following query:

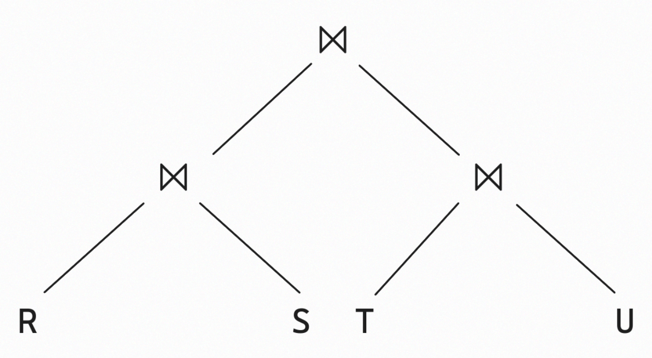
SELECT id FROM Hotels WHERE price >= 60 AND price < 100

Calculate the cost estimate for this query…

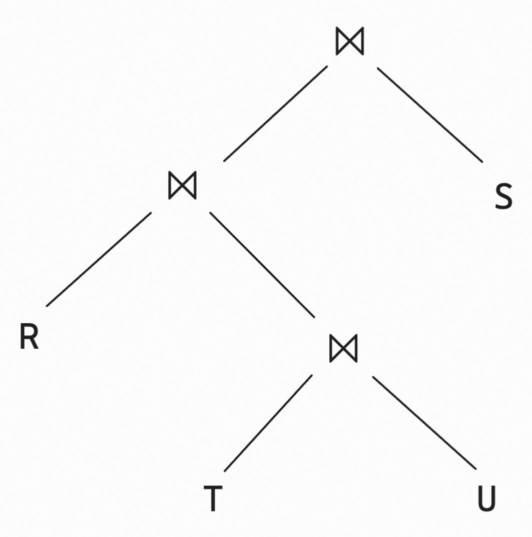
1. with a file scan
2. with a B+ Tree index on price, stored in 50 pages
    a. that is *clustered*
    b. that is *unclustered*
3. with a *hash index* on (id, price), stored in 100 pages

Question 8: Consider a query that involves joining four relations R, S, T and U. Which of the following join strategies will **never** be chosen by the System R query optimizer, given that all the options can produce a valid result set?

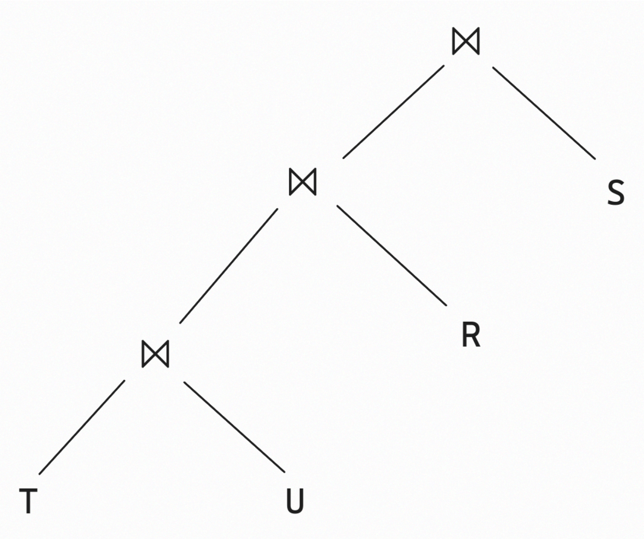
A.



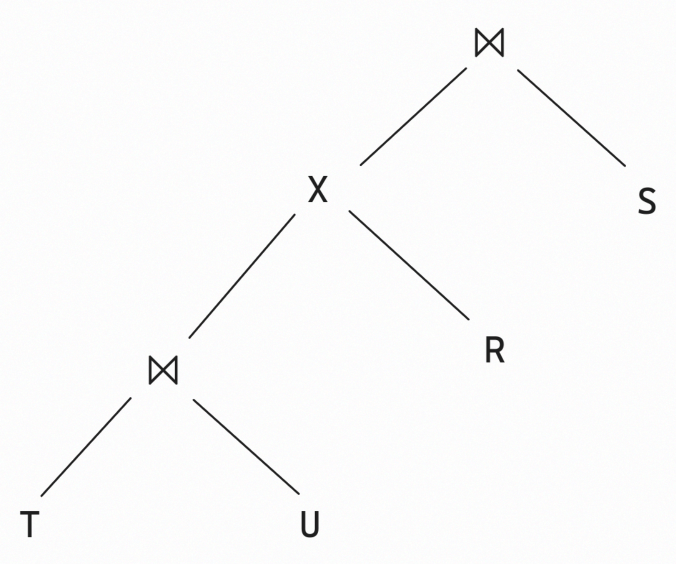
B.

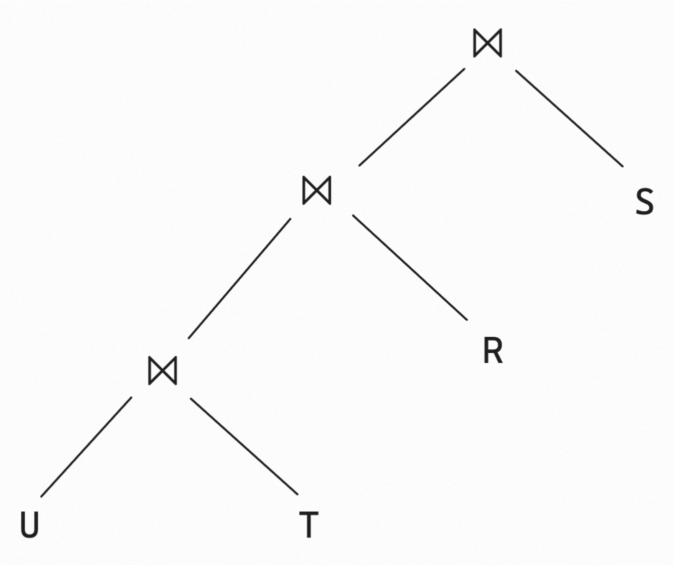


C.



D.



E. 

Question 9: Consider the following query:

SELECT R.name, S.name, T.name, U.name FROM R, S, T, U
WHERE
R.sid = S.sid AND
R.tid = T.tid AND
T.tid = U.tid

Suppose that we have the following joins at Pass 2 of the System R optimizer:

- 1. R ⋈ S with NLJ, cost estimate = 1500
- 2. R ⋈ T with SMJ, cost estimate = 5000
- 3. R ⋈ U with NLJ, cost estimate = 2000
- 4. T ⋈ U with index NLJ, cost estimate = 3000

Which of the joins will be kept by the optimizer into pass 3?

A. 1 and 4
B. 1, 2 and 4
C. All joins
D. Only 1

Extra exercises 8: Query optimization
Solutions

Answer 1: true
Answer 2: false (we can drop all projections but the **outermost** one).
Answer 3: true
Answer 4: false (it often contains approximation and isn't always up to date)
Answer 5: B (est_size = NTuples(R)*NTuples(S)/MAX{NKeys(A,S), NKeys(A,R)})
Answer 6: A (projection removed the join column).
Answer 7:
Calculate the cost estimate for this query

1. with a file scan: 1000 (# of pages)
2. with a B+ Tree index on price, stored in 50 pages…
   a. that is *clustered*: 210 = (1000) * (1 / 5) = (# of pages) * RF
   b. that is *unclustered:* 10010 = (50 + 50000) * (1 / 5) = (#index + #rec R) * RF
3. with a *hash index* on (id, price), stored in 100 pages: 100 (#index, index-only scan)

Answer 8: A + B (not all left) + D (cross-joins)

Answer 9: B

- A has the lowest cost, so it is kept.
- B and D has "interesting order" (tid is used for joining R, T and U), so both are kept
- C is cheaper than B and D, but has no interesting order, so it is removed.