How does a query tree represent a relational algebra expression? What is meant by an execution of a query tree? Discuss the rules for transformation of query trees and identify when each rule should be applied during optimization.

ANS 5)

A query tree is a tree data structure that corresponds to a relational algebra expression. It represents the input relations of the query as leaf nodes of the tree and represents the relational algebra operations as internal nodes.

An execution of the query tree consists of executing an internal node operation whenever its operands (represented by its child nodes) are available, and then replacing that internal node by the relation that results from executing the operation. The execution terminates when the root node is executed and produces the result relation for the query.

Rules for transformation of query trees:

1. Cascade of σ A conjunctive selection condition can be broken up into a cascade (that is, a sequence) of individual σ operations:



1. Commutativity of σ. The σ operation is commutative:

Text

Description automatically generated

1. Cascade of π. In a cascade (sequence) of π operations, all but the last one can be ignored:



1. Commuting σ with π. If the selection condition c involves only those attributes A1, . . ., An in the projection list, the two operations can be commuted:



1. Commutativity of ⋈ (and ×). The join operation is commutative, as is the × operation:

Text

Description automatically generated

Notice that although the order of attributes may not be the same in the relations resulting from the two joins (or two Cartesian products), the meaning is the same because the order of attributes is not important in the alternative definition of relation.

1. Commuting σ with ⋈ (or ×). If all the attributes in the selection condition c involve only the attributes of one of the relations being joined—say, R—the two operations can be commuted as follows:



Alternatively, if the selection condition c can be written as (c1 AND c2), where condition c1 involves only the attributes of R and condition c2 involves only the attributes of S, the operations commute as follows:

A picture containing text

Description automatically generated

The same rules apply if the ⋈ is replaced by a × operation.

1. Commuting π with ⋈ (or ×). Suppose that the projection list is L = {A1, ..., An, B1, ..., Bm}, where A1, ..., An are attributes of R and B1, ..., Bm are attributes of S. If the join condition c involves only attributes in L, the two operations can be commuted as follows:



If the join condition c contains additional attributes not in L, these must be added to the projection list, and a final π operation is needed. For example, if attributes An+1, ..., An+k of R and Bm+1, ..., Bm+p of S are involved in the join condition c but are not in the projection list L, the operations commute as follows:



For ×, there is no condition c, so the first transformation rule always applies by replacing c with ×.

1. Commutativity of set operations. The set operations ∪ and ∩ are commutative but − is not.
2. Associativity of ⋈, ×, ∪, and ∩. These four operations are individually associative; that is, if θ stands for any one of these four operations (throughout the expression), we have:
3. Commuting σ with set operations. The σ operation commutes with ∪, ∩, and −. If θ stands for any one of these three operations (throughout the expression), we have:

Text

Description automatically generated with medium confidence

1. The π operation commutes with ∪.



1. Converting a (σ, ×) sequence into ⋈. If the condition c of a σ that follows a × corresponds to a join condition, convert the (σ, ×) sequence into a ⋈ as follows:



There are other possible transformations. For example, a selection or join condition c can be converted into an equivalent condition by using the following standard rules from Boolean algebra (DeMorgan’s laws):

Text

Description automatically generated with medium confidence