Chapter 13: Query Optimization
Chapter 13: Query Optimization

- Introduction
- Transformation of Relational Expressions
- Catalog Information for Cost Estimation
- Statistical Information for Cost Estimation
- Cost-based optimization
- Dynamic Programming for Choosing Evaluation Plans
- Materialized views
A given query can be evaluated in alternative ways by using:

- Equivalent expressions
- Different algorithms for each operation

Consider the following relational-algebra expression, for the query “Find the names of all instructors in the Music department together with the course title of all the courses that the instructors teach.”

\[ \Pi_{name, title} (\sigma_{dept\_name = \text{"Music"}} (instructor \bowtie (teaches \bowtie \Pi_{course\_id, title}(course)))) \]

Can you find an expression that returns the same result?
Solution

A given query can be evaluated in alternative ways by using:
- Equivalent expressions
An evaluation plan defines what algorithm is used for each operation, and how the execution of the operations is coordinated.

\[ \Pi_{\text{name, title}} \text{(sort to remove duplicates)} \]

\[ (\text{hash join)} \]

\[ (\text{merge join)} \]

\[ \sigma_{\text{dept}_\text{name} = \text{Music}} \text{(use index 1)} \]

\[ \sigma_{\text{year} = 2009} \text{(use linear scan)} \]

\[ \text{pipeline} \]

\[ \text{instructor} \]

\[ \text{teaches} \]
Cost-based query optimization

- Cost difference between evaluation plans for a query can be enormous
  - E.g. seconds vs. days in some cases
- The DBMS implements cost-based query optimization
  1. Generate logically equivalent expressions using equivalence rules
  2. Annotate resultant expressions to get alternative query plans
  3. Choose the cheapest plan based on estimated cost
Cost-based query optimization

Estimation of plan cost is based on:

- Statistical information about relations. Examples:
  - nr. of tuples
  - nr. of distinct values for an attribute
  - distribution of values (e.g. uniform, normal)
- Statistics estimation for intermediate results
  - to compute cost of complex expressions
- Cost formulae for algorithms, computed using statistics. Examples:
  - A1 through A6
  - Merge-sort
  - Join algorithms
13.2 Transformation of Relational Expressions

Two relational algebra expressions are said to be equivalent if the two expressions generate the same set of tuples on every legal database instance.

Notes:

- The order of the tuples is irrelevant!
- We don’t care if they generate different results on databases that violate the integrity constraints!

In SQL, inputs and outputs are multisets of tuples:

- Two expressions in the multiset version of the relational algebra are said to be equivalent if the two expressions generate the same multiset of tuples on every legal database instance.

An equivalence rule says that expressions of two forms are equivalent:

- Can replace expression of first form by second, or vice versa.
Equivalence Rules

1. Conjunctive selection operations can be deconstructed into a sequence of individual selections.
   \[ (E) = \bigcap_{1}^{2} (E) \]

2. Selection operations are commutative.
   \[ (E) = \bigcap_{1}^{2} (E) \]

3. Only the last in a sequence of projection operations is needed, the others can be omitted.
   \[ \Pi_{L_1} (\Pi_{L_2} (\ldots (\Pi_{L_n} (E)) \ldots)) = \Pi_{L_1} (E) \]

4. Selections can be combined with Cartesian products and theta joins.
   a. \[ \sigma_{\theta}(E_1 \times E_2) = E_1 \bowtie_{\theta} E_2 \]
   b. \[ \sigma_{\theta_1}(E_1 \bowtie_{\theta_2} E_2) = E_1 \bowtie_{\theta_1 \wedge \theta_2} E_2 \]
Equivalence Rules (Cont.)

5. Theta-join operations (and natural joins) are commutative.
   \[ E_1 \bowtie_\theta E_2 = E_2 \bowtie_\theta E_1 \]

6. (a) Natural join operations are associative:
   \[ (E_1 \bowtie E_2) \bowtie E_3 = E_1 \bowtie (E_2 \bowtie E_3) \]

(b) Theta joins are associative in the following manner:
\[ (E_1 \bowtie_{\theta_1} E_2) \bowtie_{\theta_2 \land \theta_3} E_3 = E_1 \bowtie_{\theta_1 \land \theta_3} (E_2 \bowtie_{\theta_2} E_3) \]

where \( \theta_2 \) involves attributes from only \( E_2 \) and \( E_3 \).
7. The selection operation distributes over the theta join operation under the following two conditions:

(a) When all the attributes in $\theta_0$ involve only the attributes of one of the expressions ($E_1$) being joined.

$$\sigma_{\theta_0}(E_1 \Join \theta E_2) = (\sigma_{\theta_0}(E_1)) \Join \theta E_2$$

(b) When $\theta_1$ involves only the attributes of $E_1$ and $\theta_2$ involves only the attributes of $E_2$.

$$\sigma_{\theta_1 \land \theta_2}(E_1 \Join \theta E_2) = (\sigma_{\theta_1}(E_1)) \Join \theta (\sigma_{\theta_2}(E_2))$$
Pictorial Depiction of Equivalence Rules

Rule 5

Rule 6a

Rule 7a

If $\theta$ only has attributes from $E_1$
Equivalence Rules (Cont.)

8. The projection operation distributes over the theta join operation as follows:

(a) if $\theta$ involves only attributes from $L_1 \cup L_2$:

$$\Pi_{L_1 \cup L_2}(E_1 \bowtie_{\theta} E_2) = (\Pi_{L_1}(E_1)) \bowtie_{\theta} (\Pi_{L_2}(E_2))$$

(b) Consider a join $E_1 \bowtie_{\theta} E_2$.

- Let $L_1$ and $L_2$ be sets of attributes from $E_1$ and $E_2$, respectively.
- Let $L_3$ be attributes of $E_1$ that are involved in join condition $\theta$, but are not in $L_1 \cup L_2$, and
- let $L_4$ be attributes of $E_2$ that are involved in join condition $\theta$, but are not in $L_1 \cup L_2$.

$$\Pi_{L_1 \cup L_2}(E_1 \bowtie_{\theta} E_2) = \Pi_{L_1 \cup L_2}((\Pi_{L_1 \cup L_3}(E_1)) \bowtie_{\theta} (\Pi_{L_2 \cup L_4}(E_2)))$$
QUIZ: Draw the tree representations!

8. The projection operation distributes over the theta join operation as follows:

(a) if \( \theta \) involves only attributes from \( L_1 \cup L_2 \):

\[
\Pi_{L_1 \cup L_2} (E_1 \Join_\theta E_2) = (\Pi_{L_1} (E_1)) \Join_\theta (\Pi_{L_2} (E_2))
\]
8. The projection operation distributes over the theta join operation as follows:

(a) if \( \theta \) involves only attributes from \( L_1 \cup L_2 \):

\[
\Pi_{L_1 \cup L_2} (E_1 \bowtie_{\theta} E_2) = (\Pi_{L_1} (E_1)) \bowtie_{\theta} (\Pi_{L_2} (E_2))
\]

This is informally known as “pushing projection through a theta-join.”
Equivalence Rules (Cont.)

9. The set operations union and intersection are commutative
\[ E_1 \cup E_2 = E_2 \cup E_1 \]
\[ E_1 \cap E_2 = E_2 \cap E_1 \]
(set difference is not commutative).

10. Set union and intersection are associative.
\[ (E_1 \cup E_2) \cup E_3 = E_1 \cup (E_2 \cup E_3) \]
\[ (E_1 \cap E_2) \cap E_3 = E_1 \cap (E_2 \cap E_3) \]

11. The selection operation distributes over \( \cup \), \( \cap \) and \( - \).
\[ \sigma_\theta (E_1 - E_2) = \sigma_\theta (E_1) - \sigma_\theta(E_2) \]
and similarly for \( \cup \) and \( \cap \) in place of \( - \)
Also:
\[ \sigma_\theta (E_1 - E_2) = \sigma_\theta(E_1) - E_2 \]
and similarly for \( \cap \) in place of \( - \), but not for \( \cup \)

12. The projection operation distributes over union
\[ \Pi_L(E_1 \cup E_2) = (\Pi_L(E_1)) \cup (\Pi_L(E_2)) \]
Transformation Example: Pushing Selections

Query: Find the names of all instructors in the Music department, along with the titles of the courses that they teach

\[ \Pi_{\text{name, title}} \left( \sigma_{\text{dept\_name} = \text{“Music”}} \left( \text{instructor} \bowtie \left( \text{teaches} \bowtie \Pi_{\text{course\_id, title}} \left( \text{course} \right) \right) \right) \right) \]

Transformation using rule 7a.

7. The selection operation distributes over the theta join operation under the following two conditions:
   (a) When all the attributes in \( \theta_0 \) involve only the attributes of one of the expressions (\( E_1 \)) being joined.

   \[ \sigma_{\theta_0}(E_1 \bowtie \theta E_2) = (\sigma_{\theta_0}(E_1)) \bowtie \theta E_2 \]

\[ \Pi_{\text{name, title}} \left( \left( \sigma_{\text{dept\_name} = \text{“Music”}} \left( \text{instructor} \right) \right) \bowtie \left( \text{teaches} \bowtie \Pi_{\text{course\_id, title}} \left( \text{course} \right) \right) \right) \]

Performing the selection as early as possible reduces the size of the relation to be joined!
QUIZ: Draw the tree representations!

Query: Find the names of all instructors in the Music department, along with the titles of the courses that they teach

\[ \Pi_{\text{name, title}}(\sigma_{\text{dept\_name}= \text{"Music"}}(\text{instructor} \bowtie (\text{teaches} \bowtie \Pi_{\text{course\_id, title}}(\text{course}))))) \]

Transformation using rule 7a.

7. The selection operation distributes over the theta join operation under the following two conditions:
(a) When all the attributes in \( \theta_0 \) involve only the attributes of one of the expressions \( (E_1) \) being joined.

\[ \sigma_{\theta_0}(E_1 \bowtie \theta E_2) = (\sigma_{\theta_0}(E_1)) \bowtie \theta E_2 \]

\[ \Pi_{\text{name, title}}((\sigma_{\text{dept\_name}= \text{"Music"}}(\text{instructor})) \bowtie (\text{teaches} \bowtie \Pi_{\text{course\_id, title}}(\text{course})))) \]

Performing the selection as early as possible reduces the size of the relation to be joined!
Example with Multiple Transformations

Query: Find the names of all instructors in the Music department who have taught a course in 2009, along with the titles of the courses that they taught

\[
\Pi_{\text{name}, \text{title}}(\sigma_{\text{dept}_\text{name} = \text{"Music"} \land \text{year} = 2009} (\text{instructor} \bowtie (\text{teaches} \bowtie \Pi_{\text{course}_\text{id}, \text{title}}(\text{course})))))
\]

Transformation using join associatively (Rule 6a):

\[
\Pi_{\text{name}, \text{title}}(\sigma_{\text{dept}_\text{name} = \text{"Music"} \land \text{year} = 2009} ((\text{instructor} \bowtie \text{teaches}) \bowtie \Pi_{\text{course}_\text{id}, \text{title}}(\text{course}))))
\]

Second form provides an opportunity to apply the “perform selections early” rule, resulting in the subexpression

\[
\sigma_{\text{dept}_\text{name} = \text{"Music"}} (\text{instructor}) \bowtie \sigma_{\text{year} = 2009} (\text{teaches})
\]
Evaluation trees for previous example

(a) Initial expression tree

(b) Tree after multiple transformations
Transformation Example: Pushing Projections

Consider: \( \Pi_{name, title} (\sigma_{dept\_name=\text{"Music"}} (instructor) \bowtie teaches) \bowtie \Pi_{course\_id, title} (course)) \)

When we compute
\( (\sigma_{dept\_name=\text{"Music"}} (instructor) \bowtie teaches) \)
we obtain a relation whose schema is:
(\( ID, name, dept\_name, salary, course\_id, sec\_id, semester, year \))

Push projections using equivalence rules 8a and 8b; eliminate unneeded attributes from intermediate results to get:
\( \Pi_{name, title} (\Pi_{name, course\_id} ( \sigma_{dept\_name=\text{"Music"}} (instructor) \bowtie teaches) \bowtie \Pi_{course\_id, title} (course)))) \)

Performing the projection as early as possible reduces the size of the relation to be joined.
QUIZ: Draw the tree representations!

- Consider: \( \Pi_{\text{name}, \text{title}}(\sigma_{\text{dept}\_\text{name}= \text{"Music"}}(\text{instructor} \bowtie \text{teaches}) \bowtie \Pi_{\text{course}\_\text{id}, \text{title}}(\text{course}))) \)

- When we compute
  \( (\sigma_{\text{dept}\_\text{name}= \text{"Music"}}(\text{instructor} \bowtie \text{teaches}) \bowtie \Pi_{\text{course}\_\text{id}, \text{title}}(\text{course}))) \)

  we obtain a relation whose schema is:
  \(( \text{ID}, \text{name}, \text{dept}\_\text{name}, \text{salary}, \text{course}\_\text{id}, \text{sec}\_\text{id}, \text{semester}, \text{year})\)

- Push projections using equivalence rules 8a and 8b; eliminate unneeded attributes from intermediate results to get:
  \( \Pi_{\text{name}, \text{title}}(\Pi_{\text{name}, \text{course}\_\text{id}}(\sigma_{\text{dept}\_\text{name}= \text{"Music"}}(\text{instructor} \bowtie \text{teaches}) \bowtie \Pi_{\text{course}\_\text{id}, \text{title}}(\text{course})))) \)

- Performing the projection as early as possible reduces the size of the relation to be joined.
\[ \Pi_{\text{name}, \text{title}}(\sigma_{\text{dept\_name}=\text{"Music"}}(\text{instructor}) \bowtie \text{teaches}) \bowtie \Pi_{\text{course\_id}, \text{title}}(\text{course})) ) \]

\[ \Pi_{\text{name}, \text{title}}(\Pi_{\text{name}, \text{course\_id}}(\sigma_{\text{dept\_name}=\text{"Music"}}(\text{instructor}) \bowtie \text{teaches}) \bowtie \Pi_{\text{course\_id}, \text{title}}(\text{course})) ) \]
Join Ordering Example

- For all relations \( r_1, r_2, \) and \( r_3 \),
  \[
  (r_1 \bowtie r_2) \bowtie r_3 = r_1 \bowtie (r_2 \bowtie r_3)
  \]
  (Join Associativity)

- If \( r_2 \bowtie r_3 \) is quite large and \( r_1 \bowtie r_2 \) is small, we choose
  
  \[
  (r_1 \bowtie r_2) \bowtie r_3
  \]
  so that we compute and store a smaller temporary relation.
Join Ordering Example (Cont.)

- Consider the expression

$$
\Pi_{name, title}(\sigma_{dept\_name=\text{"Music"}}(instructor) \bowtie teaches) 
\bowtie \Pi_{course\_id, title}(course)))
$$

- Could compute $teaches \bowtie \Pi_{course\_id, title}(course)$ first, and join result with

$$
\sigma_{dept\_name=\text{"Music"}}(instructor)
$$

but the result of the first join is likely to be a large relation.

- Only a small fraction of the university’s instructors are likely to be from the Music department
  - it is better to compute

$$
\sigma_{dept\_name=\text{"Music"}}(instructor) \bowtie teaches
$$

first.
Enumeration of Equivalent Expressions

- Query optimizers use equivalence rules to **systematically** generate expressions equivalent to the given expression.
- Can generate all equivalent expressions as follows:
  - Repeat
    - apply all applicable equivalence rules on every subexpression of every equivalent expression found so far
    - add newly generated expressions to the set of equivalent expressions
  Until no new equivalent expressions are generated above.
- The above approach is very expensive in space and time.
  - Two approaches
    - Optimized plan generation based on transformation rules
    - Special case approach for queries with only selections, projections and joins.
13.3 Statistical Information for Cost Estimation

- \( n_r \): number of tuples in a relation \( r \).
- \( b_r \): number of blocks containing tuples of \( r \).
- \( l_r \): size of a tuple of \( r \).
- \( f_r \): blocking factor of \( r \) — i.e., the number of tuples of \( r \) that fit into one block.
- \( V(A, r) \): number of distinct values that appear in \( r \) for attribute \( A \); same as the size of \( \Pi_A(r) \).
- If tuples of \( r \) are stored together physically in a file, then:

\[
b_r = \frac{n_r}{f_r}
\]
Histograms

Example: Histogram on attribute *age* of relation *person*

- **Equi-width** histograms
- **Equi-depth** histograms

Which type is the one pictured above?
13.3.2 Selection Size Estimation

- $\sigma_{A=\vee}(r)$
  - $n_r / V(A,r)$: number of records that will satisfy the selection
  - Equality condition on a key attribute: size estimate = 1

- $\sigma_{A\leq\vee}(r)$ (case of $\sigma_{A\geq\vee}(r)$ is symmetric)

Let $c$ denote the estimated number of tuples satisfying the condition.

- If min$(A,r)$ and max$(A,r)$ are available in catalog, we assume **Uniform** distribution:
  - $c = 0$ if $v < \min(A,r)$
  - $c = \left(\frac{v \cdot \min(A,r)}{\max(A,r) \cdot \min(A,r)}\right)$

- We can use more complex theoretical distributions, e.g. **Normal**
- If histograms are available, we can use them instead of theoretical distr.
- In absence of statistical information $c$ is assumed to be $n_r / 2$. 
READ:

Size Estimation of Complex Selections
-- Conjunction (AND)
-- Disjunction (OR)
13.3.3 Join Size Estimation

Numerical example for this section

student \(\times\) takes

Catalog information:

- \(n_{student} = 5,000\)
- \(f_{student} = 50\), which implies that \(b_{student} = 5000/50 = 100\).
- \(n_{takes} = 10000\)
- \(f_{takes} = 25\), which implies that \(b_{takes} = 10000/25 = 400\)
- \(V(ID, takes) = 2500\), which implies that on average, each student who has taken a course has taken 4 courses
  - Attribute ID in takes is a foreign key referencing student
- \(V(ID, student) = 5000\) (primary key!)
13.3.3 Join Size Estimation

1. Cartesian product:
   - $r \times s$ contains $n_r \cdot n_s$ tuples; each tuple occupies $s_r + s_s$ bytes.

2. Natural join:
   - If $R \cap S = \emptyset$, then $r \bowtie s$ is the same as $r \times s$.
   - If $R \cap S$ is a key for $R$, then a tuple of $s$ will join with at most one tuple from $r$
     - therefore, the number of tuples in $r \bowtie s$ is no greater than the number of tuples in $s$.
     - The case $R \cap S$ being a key for $S$ is symmetric.
   - If $R \cap S$ in $S$ is a foreign key in $S$ referencing $R$, then the number of tuples in $r \bowtie s$ is exactly the same as the number of tuples in $s$.
     - The case for $R \cap S$ being a foreign key referencing $S$ is symmetric.
Example: Join with FK

In the query \( \text{student} \bowtie \text{takes} \), \( ID \) in \( \text{takes} \) is a foreign key referencing \( \text{student} \)

- hence, the result has exactly \( n_{\text{takes}} \) tuples, which is 10000.
Join Size Estimation (cont.)

2. Natural join (cont.)

- If \( R \cap S = \{A\} \) is not a key for \( R \) or \( S \).

  If we assume that every tuple \( t \) in \( R \) produces tuples in \( R \bowtie S \), the number of tuples in \( R \bowtie S \) is estimated to be:

  \[
  \frac{n_r \times n_s}{\text{\( V(A,s) \)}}
  \]

  If the reverse is true, the estimate obtained will be:

  \[
  \frac{n_r \times n_s}{\text{\( V(A,r) \)}}
  \]

  The lower of these two estimates is probably the more accurate one.

- Can improve on above if histograms are available

  - Use formula similar to above, for each cell of histograms on the two relations
Example: Join with no keys or FKS

Compute the size estimates for student \( \times \) takes without using information about any keys:

- \( V(\text{ID}, \text{takes}) = 2500, \) and \( V(\text{ID}, \text{student}) = 5000 \)
- The two estimates are \( 5000 \times \frac{10000}{2500} = 20,000 \) and \( 5000 \times \frac{10000}{5000} = 10000 \)
- We choose the lower estimate, which in this case, is the same as our earlier computation using foreign keys.
Join Size Estimation (cont.)

3. Theta join

- \( r \bowtie_\theta s = \) can be rewritten as \( \sigma_\theta (r \times s) \)

- Now we can apply the previous estimates for:
  - Cartesian product
  - Selection
Size Estimation for Other Operations

- Projection: estimated size of $\Pi_A(r) = V(A,r)$
- Aggregation: estimated size of $\mathcal{A}g_F(r) = V(A,r)$
- Set operations
  - For unions/intersections of selections on the same relation: rewrite and use size estimate for selections
    - E.g. $\sigma_{\theta_1}(r) \cup \sigma_{\theta_2}(r)$ can be rewritten as $\sigma_{\theta_1 \vee \theta_2}(r)$
  - For operations on different relations:
    - estimated size of $r \cup s = \text{size of } r + \text{size of } s.$
    - estimated size of $r \cap s = \text{minimum size of } r \text{ and size of } s.$
    - estimated size of $r - s = r$.
    - All the three estimates may be quite inaccurate, but provide upper bounds on the sizes.
Size Estimation for Other Operations (cont.)

- Outer join:
  - Estimated size of \( r \bowtie s = \text{size of } r \bowtie s + \text{size of } r \)
    - Case of right outer join is symmetric
  - Estimated size of \( r \bowtie s = \text{size of } r \bowtie s + \text{size of } r + \text{size of } s \)
Estimation of Number of Distinct Values

\[ V(A, r) \]

1. Selection: \( \sigma_\theta (r) \)
   - If \( \theta \) forces \( A \) to take a specified value: \( V(A, \sigma_\theta (r)) = 1 \).
     - e.g., \( A = 3 \)
   - If \( \theta \) forces \( A \) to take on one of a specified set of values:
     \[ V(A, \sigma_\theta (r)) = \text{number of specified values} \]
     - (e.g., (\( A = 1 \ V A = 3 \ V A = 4 \ ))
   - If the selection condition \( \theta \) is of the form \( A \ op r \)
     estimated \( V(A, \sigma_\theta (r)) = V(A.r) * s \)
     - where \( s \) is the selectivity of the selection.
   - In all the other cases: use approximate estimate of
     \[ \min(V(A,r), n_{\sigma_\theta (r)}) \]
     - More accurate estimate can be got using probability theory, but this one works fine generally.
Estimation of Number of Distinct Values $V(A, r)$

2. Join: $r \bowtie s$

- If all attributes in $A$ are from $r$
  
  
  
  
  estimated $V(A, r \bowtie s) = \min (V(A,r), n_{r \bowtie s})$

- If $A$ contains attributes $A_1$ from $r$ and $A_2$ from $s$, then estimated $V(A, r \bowtie s) =$

  \[
  \min(V(A_1,r)^* V(A_2 - A_1,s), V(A_1 - A_2,r)^* V(A_2,s), n_{r \bowtie s})
  \]

- More accurate estimate can be obtained using probability distributions, but this one works fine generally

3. Etc. etc.
13.4 Choice of evaluation plans

Cost-based Join Order Selection

For a complex join query, the number of different query plans that are equivalent to the query can be large. As an illustration, consider the expression:

$$r_1 \bowtie r_2 \bowtie \cdots \bowtie r_n$$

where the joins are expressed without any ordering. With $n = 3$, there are 12 different join orderings:

- $r_1 \bowtie (r_2 \bowtie r_3)$
- $r_1 \bowtie (r_3 \bowtie r_2)$
- $(r_2 \bowtie r_3) \bowtie r_1$
- $(r_3 \bowtie r_2) \bowtie r_1$
- $r_2 \bowtie (r_1 \bowtie r_3)$
- $r_2 \bowtie (r_3 \bowtie r_1)$
- $(r_1 \bowtie r_3) \bowtie r_2$
- $(r_3 \bowtie r_1) \bowtie r_2$
- $r_3 \bowtie (r_1 \bowtie r_2)$
- $r_3 \bowtie (r_2 \bowtie r_1)$
- $(r_1 \bowtie r_2) \bowtie r_3$
- $(r_2 \bowtie r_1) \bowtie r_3$
SKIP the remainder of this chapter, starting with p.600
Figure 13.01

(a) Initial expression tree

(b) Transformed expression tree
Figure 13.02

\[ \Pi_{\text{name, title}} (\text{merge join}) \]

\[ \Pi_{\text{course_id, title}} (\text{hash join}) \]

\[ \sigma_{\text{dept_name} = \text{Music}} (\text{use index 1}) \]

\[ \Pi_{\text{name, title}} (\text{sort to remove duplicates}) \]

\[ \text{pipeline} \]

\[ \text{sort}_{\text{ID}} \]

\[ \text{pipeline} \]

\[ \text{sort}_{\text{ID}} \]

\[ \text{pipeline} \]

\[ \text{pipeline} \]
Figure 13.03

Rule 5

If only has attributes from $E_1$

Rule 6.a

Rule 7.a
Figure 13.04

(a) Initial expression tree

(b) Tree after multiple transformations
Figure 13.08

(a) Left-deep join tree

(b) Non-left-deep join tree