• Equivalence rules
  ◦ Join commutativity
  ◦ Join associativity
• Minimal set of equivalence rules
• Enumeration of equivalent expressions
• Statistics estimation
• Catalog information
• Size estimation
  ◦ Selection
  ◦ Selectivity
  ◦ Join
• Histograms
• Distinct value estimation
• Random sample
• Choice of evaluation plans
• Interaction of evaluation techniques
• Cost-based optimization
• Join-order optimization
  ◦ Dynamic-programming algorithm
  ◦ Left-deep join order
  ◦ Interesting sort order
• Heuristic optimization
• Plan caching
• Access-plan selection
• Correlated evaluation
• Decorrelation
• Materialized views
• Materialized view maintenance
  ◦ Recomputation
  ◦ Incremental maintenance
  ◦ Insertion
  ◦ Deletion
  ◦ Updates
• Query optimization with materialized views
• Index selection
• Materialized view selection
• Top-K optimization
• Join minimization
• Halloween problem
• Multiquery optimization

Practice Exercises

13.1 Show that the following equivalences hold. Explain how you can apply them to improve the efficiency of certain queries:
   a. \( E_1 \bowtie_\theta (E_2 - E_3) = (E_1 \bowtie_\theta E_2 - E_1 \bowtie_\theta E_3) \).
   b. \( \sigma_\theta (A \cdot \Sigma E) = A \cdot \Sigma \sigma_\theta (E) \), where \( \theta \) uses only attributes from \( A \).
   c. \( \sigma_\theta (E_1 \bowtie E_2) = \sigma_\theta (E_1) \bowtie E_2 \), where \( \theta \) uses only attributes from \( E_1 \).

13.2 For each of the following pairs of expressions, give instances of relations that show the expressions are not equivalent.
   a. \( \Pi_A (R - S) \) and \( \Pi_A (R) - \Pi_A (S) \).
   b. \( \sigma_{B < 4} (A \cdot \max (B) \text{ as } B (R)) \) and \( A \cdot \max (B) \text{ as } B \cdot \sigma_{B < 4} (R) \).
c. In the preceding expressions, if both occurrences of \( \text{max} \) were replaced by \( \text{min} \) would the expressions be equivalent?

d. \((R \bowtie S) \bowtie T \) and \( R \bowtie (S \bowtie T) \)
In other words, the natural left outer join is not associative. (Hint: Assume that the schemas of the three relations are \( R(a, b1) \), \( S(a, b2) \), and \( T(a, b3) \), respectively.)

e. \( \sigma_{\theta}(E_1 \bowtie E_2) \) and \( E_1 \bowtie \sigma_{\theta}(E_2) \), where \( \theta \) uses only attributes from \( E_2 \).

13.3 SQL allows relations with duplicates (Chapter 3).

a. Define versions of the basic relational-algebra operations \( \sigma \), \( \Pi \), \( \times \), \( \bowtie \), \( - \), \( \cup \), and \( \cap \) that work on relations with duplicates, in a way consistent with SQL.

b. Check which of the equivalence rules 1 through 7.b hold for the multiset version of the relational-algebra defined in part a.

13.4 Consider the relations \( r_1(A, B, C) \), \( r_2(C, D, E) \), and \( r_3(E, F) \), with primary keys \( A, C \), and \( E \), respectively. Assume that \( r_1 \) has 1000 tuples, \( r_2 \) has 1500 tuples, and \( r_3 \) has 750 tuples. Estimate the size of \( r_1 \bowtie r_2 \bowtie r_3 \), and give an efficient strategy for computing the join.

13.5 Consider the relations \( r_1(A, B, C) \), \( r_2(C, D, E) \), and \( r_3(E, F) \) of Practice Exercise 13.4. Assume that there are no primary keys, except the entire schema. Let \( V(C, r_1) \) be 900, \( V(C, r_2) \) be 1100, \( V(E, r_2) \) be 50, and \( V(E, r_3) \) be 100. Assume that \( r_1 \) has 1000 tuples, \( r_2 \) has 1500 tuples, and \( r_3 \) has 750 tuples. Estimate the size of \( r_1 \bowtie r_2 \bowtie r_3 \) and give an efficient strategy for computing the join.

13.6 Suppose that a B\(^+\)-tree index on \textit{building} is available on relation \textit{department}, and that no other index is available. What would be the best way to handle the following selections that involve negation?

a. \( \sigma_{\neg \text{building} < \textit{"Watson"}} \)(\textit{department})

b. \( \sigma_{\neg \text{building} = \textit{"Watson"}} \)(\textit{department})

c. \( \sigma_{\neg \text{building} < \textit{"Watson"} \lor \text{budget} < 50000} \)(\textit{department})

13.7 Consider the query:

```sql
select *
from r, s
where upper(r.A) = upper(s.A);
```

where “upper” is a function that returns its input argument with all lowercase letters replaced by the corresponding uppercase letters.

a. Find out what plan is generated for this query on the database system you use.
Practice Exercises 619

b. Some database systems would use a (block) nested-loop join for this query, which can be very inefficient. Briefly explain how hash-join or merge-join can be used for this query.

13.8 Give conditions under which the following expressions are equivalent

$$A, B \sigma_{\text{agg}(C)}(E_1 \Join E_2) \quad \text{and} \quad (A \sigma_{\text{agg}(C)}(E_1)) \Join E_2$$

where \(\text{agg}\) denotes any aggregation operation. How can the above conditions be relaxed if \(\text{agg}\) is one of \text{min} or \text{max}?

13.9 Consider the issue of interesting orders in optimization. Suppose you are given a query that computes the natural join of a set of relations \(S\). Given a subset \(S_1\) of \(S\), what are the interesting orders of \(S_1\)?

13.10 Show that, with \(n\) relations, there are \((2(n - 1))!/(n - 1)!\) different join orders. \textit{Hint: A complete binary tree} is one where every internal node has exactly two children. Use the fact that the number of different complete binary trees with \(n\) leaf nodes is:

$$\frac{1}{n} \left( \frac{2(n - 1)}{(n - 1)} \right)^n$$

If you wish, you can derive the formula for the number of complete binary trees with \(n\) nodes from the formula for the number of binary trees with \(n\) nodes. The number of binary trees with \(n\) nodes is:

$$\frac{1}{n + 1} \left( \frac{2n}{n} \right)^n$$

This number is known as the \textbf{Catalan number}, and its derivation can be found in any standard textbook on data structures or algorithms.

13.11 Show that the lowest-cost join order can be computed in time \(O(3^n)\). Assume that you can store and look up information about a set of relations (such as the optimal join order for the set, and the cost of that join order) in constant time. (If you find this exercise difficult, at least show the looser time bound of \(O(2^{2n})\).)

13.12 Show that, if only left-deep join trees are considered, as in the System R optimizer, the time taken to find the most efficient join order is around \(n2^n\). Assume that there is only one interesting sort order.

13.13 Consider the bank database of Figure 13.9, where the primary keys are underlined. Construct the following SQL queries for this relational database.

a. Write a nested query on the relation \textit{account} to find, for each branch with name starting with B, all accounts with the maximum balance at the branch.
620  Chapter 13  Query Optimization

branch(branch\textunderscore name, branch\textunderscore city, assets)
customer (customer\textunderscore name, customer\textunderscore street, customer\textunderscore city)
loan (loan\textunderscore number, branch\textunderscore name, amount)
borrower (customer\textunderscore name, loan\textunderscore number)
account (account\textunderscore number, branch\textunderscore name, balance)
depositor (customer\textunderscore name, account\textunderscore number)

Figure 13.9  Banking database for Exercise 13.13.

b. Rewrite the preceding query, without using a nested subquery; in other words, decorrelate the query.

c. Give a procedure (similar to that described in Section 13.4.4) for decorrelating such queries.

13.14  The set version of the semijoin operator \( \bowtie \) is defined as follows:

\[
r \bowtie_{\theta} s = \Pi_{\theta} (r \bowtie_{\theta} s)
\]

where \( R \) is the set of attributes in the schema of \( r \). The multiset version of the semijoin operation returns the same set of tuples, but each tuple has exactly as many copies as it had in \( r \).

Consider the nested query we saw in Section 13.4.4 which finds the names of all instructors who taught a course in 2007. Write the query in relational algebra using the multiset semjoin operation, ensuring that the number of duplicates of each name is the same as in the SQL query. (The semijoin operation is widely used for decorrelation of nested queries.)

Exercises

13.15  Suppose that a \( B^+ \)-tree index on \( (\text{dept\textunderscore name}, \text{building}) \) is available on relation \( \text{department} \). What would be the best way to handle the following selection?

\[
\sigma_{(\text{building} < \text{"Watson"}) \land (\text{budget} < 55000) \land (\text{dept\textunderscore name} = \text{"Music"})} (\text{department})
\]

13.16  Show how to derive the following equivalences by a sequence of transformations using the equivalence rules in Section 13.2.1.

a. \( \sigma_{\theta_1 \land \theta_2 : A_1} (E) = \sigma_{\theta_2} (\sigma_{\theta_1} (E)) \)

b. \( \sigma_{\theta_2} (E_1 \bowtie_{\theta_1} E_2) = \sigma_{\theta_2} (E_1 \bowtie_{\theta_1} (\sigma_{\theta_1} (E_2))) \), where \( \theta_2 \) involves only attributes from \( E_2 \)
13.17 Consider the two expressions $\sigma_\theta(E_1 \Join E_2)$ and $\sigma_\theta(E_1 \bowtie E_2)$.
   a. Show using an example that the two expressions are not equivalent in general.
   b. Give a simple condition on the predicate $\theta$, which if satisfied will ensure that the two expressions are equivalent.

13.18 A set of equivalence rules is said to be complete if, whenever two expressions are equivalent, one can be derived from the other by a sequence of uses of the equivalence rules. Is the set of equivalence rules that we considered in Section 13.2.1 complete? Hint: Consider the equivalence $\sigma_{3=5}(r) = \{\}$.

13.19 Explain how to use a histogram to estimate the size of a selection of the form $\sigma_{A \leq v}(r)$.

13.20 Suppose two relations $r$ and $s$ have histograms on attributes $r.A$ and $s.A$, respectively, but with different ranges. Suggest how to use the histograms to estimate the size of $r \bowtie s$. Hint: Split the ranges of each histogram further.

13.21 Consider the query

\[
\begin{align*}
\text{select} & \quad A, B \\
\text{from} & \quad r \\
\text{where} & \quad r.B < \text{some} (\text{select} B \\
& \quad \text{from} \quad s \\
& \quad \text{where} \quad s.A = r.A)
\end{align*}
\]

Show how to decorrelate the above query using the multiset version of the semijoin operation, defined in Exercise 13.14.

13.22 Describe how to incrementally maintain the results of the following operations, on both insertions and deletions:
   a. Union and set difference.
   b. Left outer join.

13.23 Give an example of an expression defining a materialized view and two situations (sets of statistics for the input relations and the differentials) such that incremental view maintenance is better than recomputation in one situation, and recomputation is better in the other situation.

13.24 Suppose you want to get answers to $r \bowtie s$ sorted on an attribute of $r$, and want only the top $K$ answers for some relatively small $K$. Give a good way of evaluating the query:
   a. When the join is on a foreign key of $r$ referencing $s$, where the foreign key attribute is declared to be not null.
   b. When the join is not on a foreign key.
13.25 Consider a relation \( r(A, B, C) \), with an index on attribute \( A \). Give an example of a query that can be answered by using the index only, without looking at the tuples in the relation. (Query plans that use only the index, without accessing the actual relation, are called index-only plans.)

13.26 Suppose you have an update query \( U \). Give a simple sufficient condition on \( U \) that will ensure that the Halloween problem cannot occur, regardless of the execution plan chosen, or the indices that exist.

Bibliographical Notes

The seminal work of Selinger et al. [1979] describes access-path selection in the System R optimizer, which was one of the earliest relational-query optimizers. Query processing in Starburst, described in Haas et al. [1989], forms the basis for query optimization in IBM DB2.

Graefe and McKenna [1993a] describe Volcano, an equivalence-rule–based query optimizer that, along with its successor Cascades (Graefe [1995]), forms the basis of query optimization in Microsoft SQL Server.

Estimation of statistics of query results, such as result size, is addressed by Ioannidis and Poosala [1995], Poosala et al. [1996], and Ganguly et al. [1996], among others. Nonuniform distributions of values cause problems for estimation of query size and cost. Cost-estimation techniques that use histograms of value distributions have been proposed to tackle the problem. Ioannidis and Christodoulakis [1993], Ioannidis and Poosala [1995], and Poosala et al. [1996] present results in this area. The use of random sampling for constructing histograms is well known in statistics, but issues in histogram construction in the context of databases is discussed in Chaudhuri et al. [1998].


Optimization of nested subqueries is discussed in Kim [1982], Ganski and Wong [1987], Dayal [1987], Seshadri et al. [1996] and Galindo-Legaria and Joshi [2001].

Blakeley et al. [1986] describe techniques for maintenance of materialized views. Optimization of materialized view maintenance plans is described by Vista [1998] and Mistry et al. [2001]. Query optimization in the presence of materialized views is addressed by Chaudhuri et al. [1995]. Index selection and materialized view selection are addressed by Ross et al. [1996], and Chaudhuri and Narasayya [1997].

Optimization of top-\( K \) queries is addressed in Carey and Kossmann [1998] and Bruno et al. [2002]. A collection of techniques for join minimization has
been grouped under the name *tableau optimization*. The notion of a tableau was introduced by Aho et al. [1979b] and Aho et al. [1979a], and was further extended by Sagiv and Yannakakis [1981].

Parametric query-optimization algorithms have been proposed by Ioannidis et al. [1992], Ganguly [1998] and Hulgeri and Sudarshan [2003]. Sellis [1988] was an early work on multiquery optimization, while Roy et al. [2000] showed how to integrate multi-query optimization into a Volcano-based query optimizer.

Galindo-Legaria et al. [2004] describes query processing and optimization for database updates, including optimization of index maintenance, materialized view maintenance plans and integrity constraint checking, along with techniques to handle the Halloween problem.