Chapter 23

Query Processing

Chapter 23 - Objectives

- Objectives of query processing and optimization.
- Static versus dynamic query optimization.
- How a query is decomposed and semantically analyzed.
- How to create a R.A.T. to represent a query.
- Rules of equivalence for RA operations.
- How to apply heuristic transformation rules to improve efficiency of a query.

Chapter 23 - Objectives

- Types of database statistics required to estimate cost of operations.
- Different strategies for implementing selection.
- How to evaluate cost and size of selection.
- Different strategies for implementing join.
- How to evaluate cost and size of join.
- Different strategies for implementing projection.
- How to evaluate cost and size of projection.

Chapter 23 - Objectives

- How to evaluate the cost and size of other RA operations.
- How pipelining can be used to improve efficiency of queries.
- Difference between materialization and pipelining.
- Advantages of left-deep trees.
- Approaches to finding optimal execution strategy.
- How Oracle handles QO.

Introduction

- In network and hierarchical DBMSs, low-level procedural query language is generally embedded in high-level programming language.
- Programmer's responsibility to select most appropriate execution strategy.
- With declarative languages such as SQL, user specifies what data is required rather than how it is to be retrieved.
- Relieves user of knowing what constitutes good execution strategy.

Introduction

- Also gives DBMS more control over system performance.

- Two main techniques for query optimization:
  - heuristic rules that order operations in a query;
  - comparing different strategies based on relative costs, and selecting one that minimizes resource usage.

- Disk access tends to be dominant cost in query processing for centralized DBMS.

Query Processing

Activities involved in retrieving data from the database.

- **Aims of QP:**
  - transform query written in high-level language (e.g. SQL), into correct and efficient execution strategy expressed in low-level language (implementing RA);
  - execute strategy to retrieve required data.

Query Optimization

Activity of choosing an efficient execution strategy for processing query.

- As there are many equivalent transformations of same high-level query, aim of QO is to choose one that minimizes resource usage.
- Generally, reduce total execution time of query.
- May also reduce response time of query.
- Problem computationally intractable with large number of relations, so strategy adopted is reduced to finding near optimum solution.

Example 21.1 - Different Strategies

Find all Managers who work at a London branch.

```
SELECT *
FROM Staff s, Branch b
WHERE s.branchNo = b.branchNo AND
(s.position = 'Manager' AND b.city = 'London');
```

Example 21.1 - Different Strategies

- Three equivalent RA queries are:

(1) $\sigma_{(position='Manager') \wedge (city='London') \wedge (Staff.branchNo=Branch.branchNo)}$ (Staff X Branch)

(2) $\sigma_{(position='Manager') \wedge (city='London')}($Staff $\bowtie_{Staff.branchNo=Branch.branchNo}$ Branch)

(3) $(\sigma_{position='Manager'}$(Staff)) $\bowtie_{Staff.branchNo=Branch.branchNo}$ $(\sigma_{city='London'}$ (Branch))

Example 21.1 - Different Strategies

- Assume:
  - 1000 tuples in Staff; 50 tuples in Branch;
  - 50 Managers; 5 London branches;
  - no indexes or sort keys;
  - results of any intermediate operations stored on disk;
  - cost of the final write is ignored;
  - tuples are accessed one at a time.

Example 21.1 - Cost Comparison

- Cost (in disk accesses) are:

(1) (1000 + 50) + 2*(1000 * 50) = 101 050

(2) 2*1000 + (1000 + 50) = 3 050

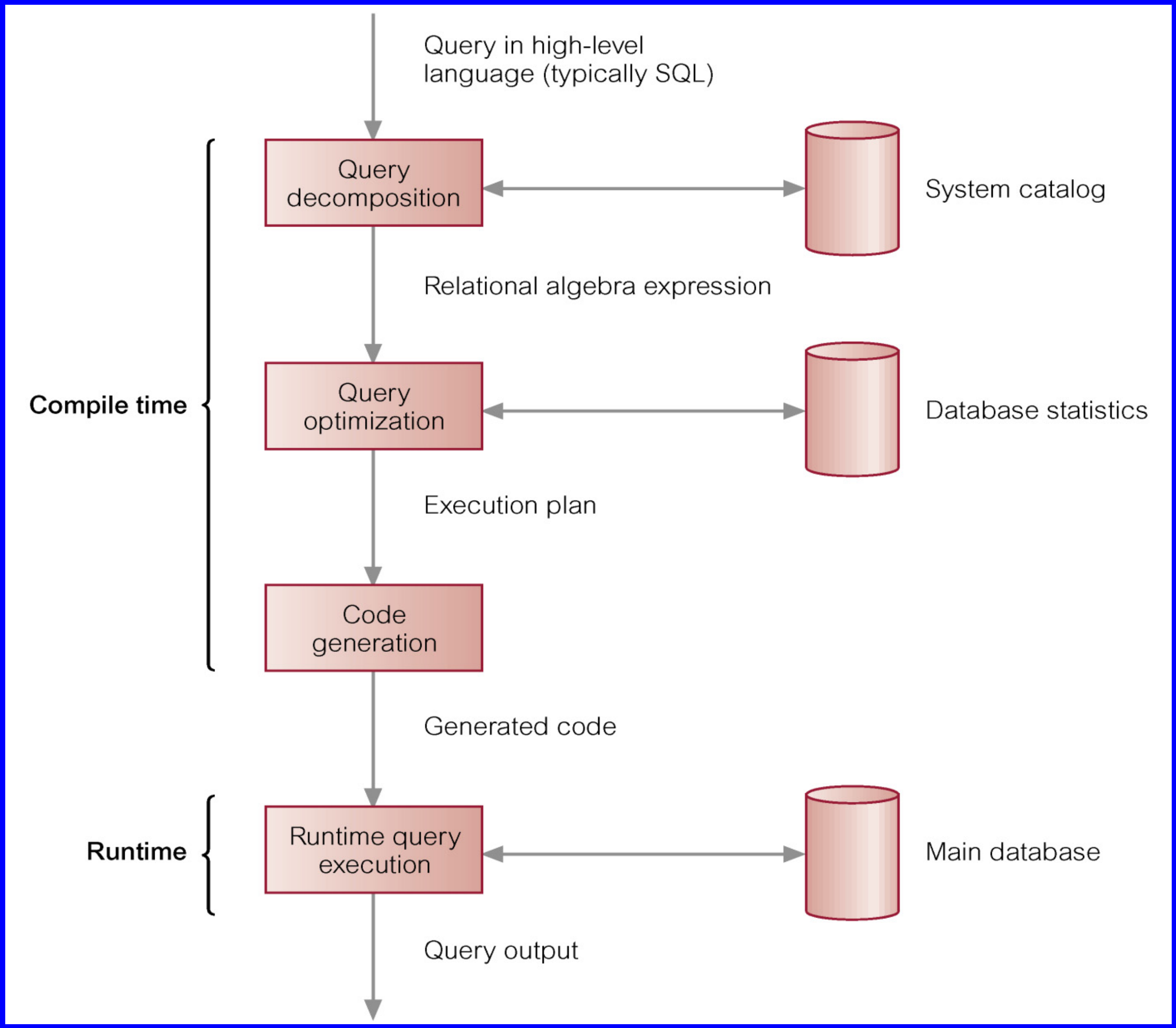
(3) 1000 + 2*50 + 5 + (50 + 5) = 1 160

- Cartesian product and join operations much more expensive than selection, and third option significantly reduces size of relations being joined together.

Phases of Query Processing

- QP has four main phases:
  - decomposition (consisting of parsing and validation);
  - optimization;
  - code generation;
  - execution.

Phases of Query Processing



Dynamic versus Static Optimization

- Two times when first three phases of QP can be carried out:
  - dynamically every time query is run;
  - statically when query is first submitted.
- Advantages of dynamic QO arise from fact that information is up to date.
- Disadvantages are that performance of query is affected, time may limit finding optimum strategy.

Dynamic versus Static Optimization

- Advantages of static QO are removal of runtime overhead, and more time to find optimum strategy.
- Disadvantages arise from fact that chosen execution strategy may no longer be optimal when query is run.
- Could use a hybrid approach to overcome this.

Query Decomposition

- Aims are to transform high-level query into RA query and check that query is syntactically and semantically correct.
- Typical stages are:
  - analysis,
  - normalization,
  - semantic analysis,
  - simplification,
  - query restructuring.

Analysis

- Analyze query lexically and syntactically using compiler techniques.
- Verify relations and attributes exist.
- Verify operations are appropriate for object type.

Analysis - Example

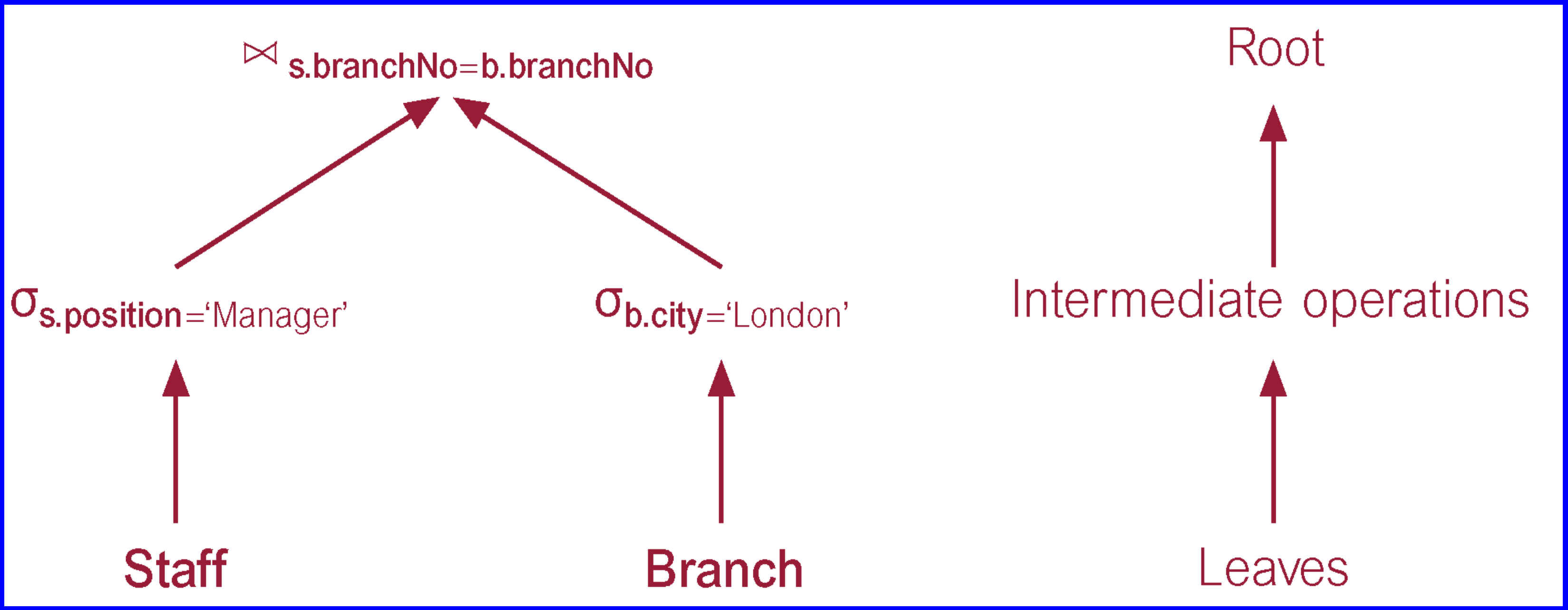
```
SELECT staff_no
FROM Staff
WHERE position > 10;
```

- This query would be rejected on two grounds:
  - staff_no is not defined for Staff relation (should be staffNo).
  - Comparison '>10' is incompatible with type position, which is variable character string.

Analysis

- Finally, query transformed into some internal representation more suitable for processing.
- Some kind of query tree is typically chosen, constructed as follows:
  - Leaf node created for each base relation.
  - Non-leaf node created for each intermediate relation produced by RA operation.
  - Root of tree represents query result.
  - Sequence is directed from leaves to root.

Example 21.1 - R.A.T.



Normalization

- Converts query into a normalized form for easier manipulation.
- Predicate can be converted into one of two forms:

Conjunctive normal form:

$$(\text{position} = \text{'Manager'} \vee \text{salary} > 20000) \wedge (\text{branchNo} = \text{'B003'})$$

Disjunctive normal form:

$$(\text{position} = \text{'Manager'} \wedge \text{branchNo} = \text{'B003'}) \vee$$
$$(\text{salary} > 20000 \wedge \text{branchNo} = \text{'B003'})$$

Simplification

- Detects redundant qualifications,
- eliminates common sub-expressions,
- transforms query to semantically equivalent but more easily and efficiently computed form.

- Typically, access restrictions, view definitions, and integrity constraints are considered.
- Assuming user has appropriate access privileges, first apply well-known idempotency rules of boolean algebra.

Transformation Rules for RA Operations

Conjunctive Selection operations can cascade into individual Selection operations (and vice versa).

$$\sigma_{p \wedge q \wedge r}(R) = \sigma_p(\sigma_q(\sigma_r(R)))$$

- **Sometimes referred to as cascade of Selection.**

$$\sigma_{branchNo='B003' \wedge salary>15000}(Staff) = \sigma_{branchNo='B003'}(\sigma_{salary>15000}(Staff))$$

Transformation Rules for RA Operations

Commutativity of Selection.

$$\sigma_p(\sigma_q(R)) = \sigma_q(\sigma_p(R))$$

- **For example:**

$$\sigma_{branchNo='B003'}(\sigma_{salary>15000}(Staff)) =$$
$$\sigma_{salary>15000}(\sigma_{branchNo='B003'}(Staff))$$

Transformation Rules for RA Operations

In a sequence of Projection operations, only the last in the sequence is required.

$$\Pi_L \Pi_M \ldots \Pi_N(R) = \Pi_L (R)$$

- For example:

$$\Pi_{lName} \Pi_{branchNo,\, lName}(Staff) = \Pi_{lName} (Staff)$$

Transformation Rules for RA Operations

Commutativity of Selection and Projection.

- If predicate p involves only attributes in projection list, Selection and Projection operations commute:

$$\Pi_{Ai, \ldots, Am}(\sigma_p(R)) = \sigma_p(\Pi_{Ai, \ldots, Am}(R))$$

$$\text{where } p \in \{A_1, A_2, \ldots, A_m\}$$

- For example:

$$\Pi_{fName, lName}(\sigma_{lName='Beech'}(Staff)) =$$
$$\sigma_{lName='Beech'}(\Pi_{fName,lName}(Staff))$$

Transformation Rules for RA Operations

Commutativity of Theta join (and Cartesian product).

$$R \bowtie_p S = S \bowtie_p R$$

$$R \times S = S \times R$$

- Rule also applies to Equijoin and Natural join. For example:

$$\text{Staff} \bowtie_{\text{staff.branchNo=branch.branchNo}} \text{Branch} =$$

$$\text{Branch} \bowtie_{\text{staff.branchNo=branch.branchNo}} \text{Staff}$$

Transformation Rules for RA Operations

Commutativity of Selection and Theta join (or Cartesian product).

- If selection predicate involves only attributes of one of join relations, Selection and Join (or Cartesian product) operations commute:

$$\sigma_p(R \bowtie_r S) = (\sigma_p(R)) \bowtie_r S$$

$$\sigma_p(R \times S) = (\sigma_p(R)) \times S$$

$$\text{where } p \in \{A_1, A_2, \ldots, A_n\}$$

Transformation Rules for RA Operations

- If selection predicate is conjunctive predicate having form $(p \wedge q)$, where p only involves attributes of R, and q only attributes of S, Selection and Theta join operations commute as:

$$\sigma_{p \wedge q}(R \bowtie_r S) = (\sigma_p(R)) \bowtie_r (\sigma_q(S))$$

$$\sigma_{p \wedge q}(R \times S) = (\sigma_p(R)) \times (\sigma_q(S))$$

Transformation Rules for RA Operations

- For example:

$$\sigma_{position='Manager' \wedge city='London'}(Staff \bowtie_{Staff.branchNo=Branch.branchNo} Branch) =$$

$$(\sigma_{position='Manager'}(Staff)) \bowtie_{Staff.branchNo=Branch.branchNo} (\sigma_{city='London'} (Branch))$$

Transformation Rules for RA Operations

Commutativity of Projection and Theta join (or Cartesian product).

- If projection list is of form $L = L_1 \cup L_2$, where L_1 only has attributes of R, and L_2 only has attributes of S, provided join condition only contains attributes of L, Projection and Theta join commute:

$$\Pi_{L1 \cup L2}(R \bowtie_r S) = (\Pi_{L1}(R)) \bowtie_r (\Pi_{L2}(S))$$

Transformation Rules for RA Operations

- **If join condition contains additional attributes not in L ($M = M_1 \cup M_2$ where M_1 only has attributes of R, and M_2 only has attributes of S), a final projection operation is required:**

$$\Pi_{L1 \cup L2}(R \bowtie_r S) = \Pi_{L1 \cup L2}((\Pi_{L1 \cup M1}(R)) \bowtie_r (\Pi_{L2 \cup M2}(S)))$$

Transformation Rules for RA Operations

- **For example:**

$$\Pi_{position,city,branchNo}(Staff \bowtie_{Staff.branchNo=Branch.branchNo} Branch) =$$
$$(\Pi_{position,\ branchNo}(Staff)) \bowtie_{Staff.branchNo=Branch.branchNo} (\Pi_{city,\ branchNo}(Branch))$$

- **and using the latter rule:**

$$\Pi_{position,\ city}(Staff \bowtie_{Staff.branchNo=Branch.branchNo} Branch) =$$
$$\Pi_{position,\ city}((\Pi_{position,\ branchNo}(Staff)) \bowtie_{Staff.branchNo=Branch.branchNo} (\Pi_{city,\ branchNo}(Branch)))$$

Transformation Rules for RA Operations

Commutativity of Union and Intersection (but not set difference).

$$R \cup S = S \cup R$$

$$R \cap S = S \cap R$$

Transformation Rules for RA Operations

Commutativity of Selection and set operations (Union, Intersection, and Set difference).

$$\sigma_p(R \cup S) = \sigma_p(S) \cup \sigma_p(R)$$

$$\sigma_p(R \cap S) = \sigma_p(S) \cap \sigma_p(R)$$

$$\sigma_p(R - S) = \sigma_p(S) - \sigma_p(R)$$

Transformation Rules for RA Operations

Commutativity of Projection and Union.

$$\Pi_L(R \cup S) = \Pi_L(S) \cup \Pi_L(R)$$

Associativity of Union and Intersection (but not Set difference).

$$(R \cup S) \cup T = S \cup (R \cup T)$$

$$(R \cap S) \cap T = S \cap (R \cap T)$$

Transformation Rules for RA Operations

Associativity of Theta join (and Cartesian product).

- Cartesian product and Natural join are always associative:

  $$(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$$

  $$(R \times S) \times T = R \times (S \times T)$$

- If join condition q involves attributes only from S and T, then Theta join is associative:

  $$(R \bowtie_{p} S) \bowtie_{q \wedge r} T = R \bowtie_{p \wedge r} (S \bowtie_{q} T)$$

Transformation Rules for RA Operations

- **For example:**

$$(\text{Staff} \bowtie_{\text{Staff.staffNo=PropertyForRent.staffNo}} \text{PropertyForRent}) \bowtie_{\text{ownerNo=Owner.ownerNo} \wedge \text{staff.lName=Owner.lName}} \text{Owner} =$$

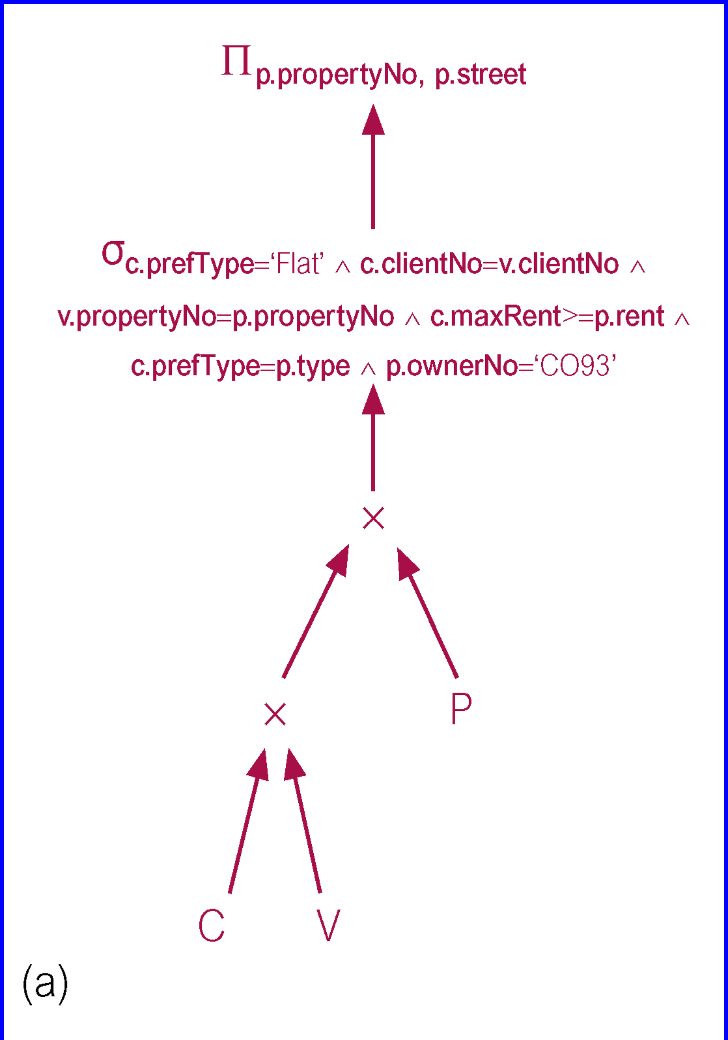
$$\text{Staff} \bowtie_{\text{staff.staffNo=PropertyForRent.staffNo} \wedge \text{staff.lName=lName}} (\text{PropertyForRent} \bowtie_{\text{ownerNo}} \text{Owner})$$

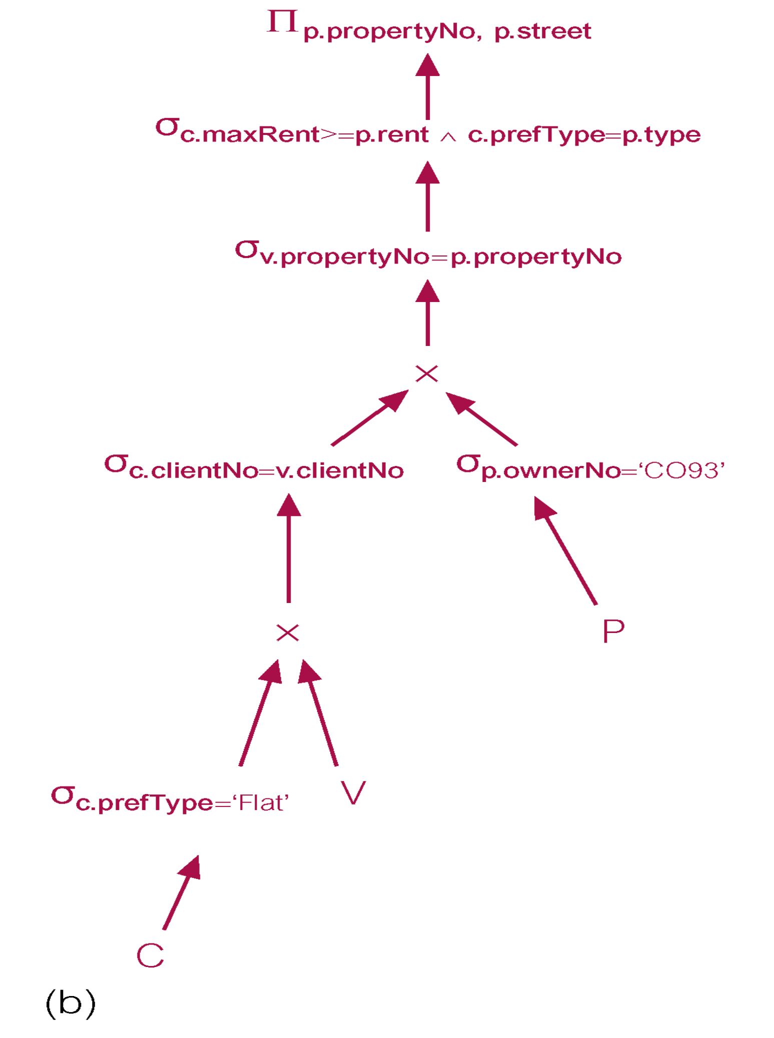
Example 21.3 Use of Transformation Rules

For prospective renters of flats, find properties that match requirements and owned by CO93.

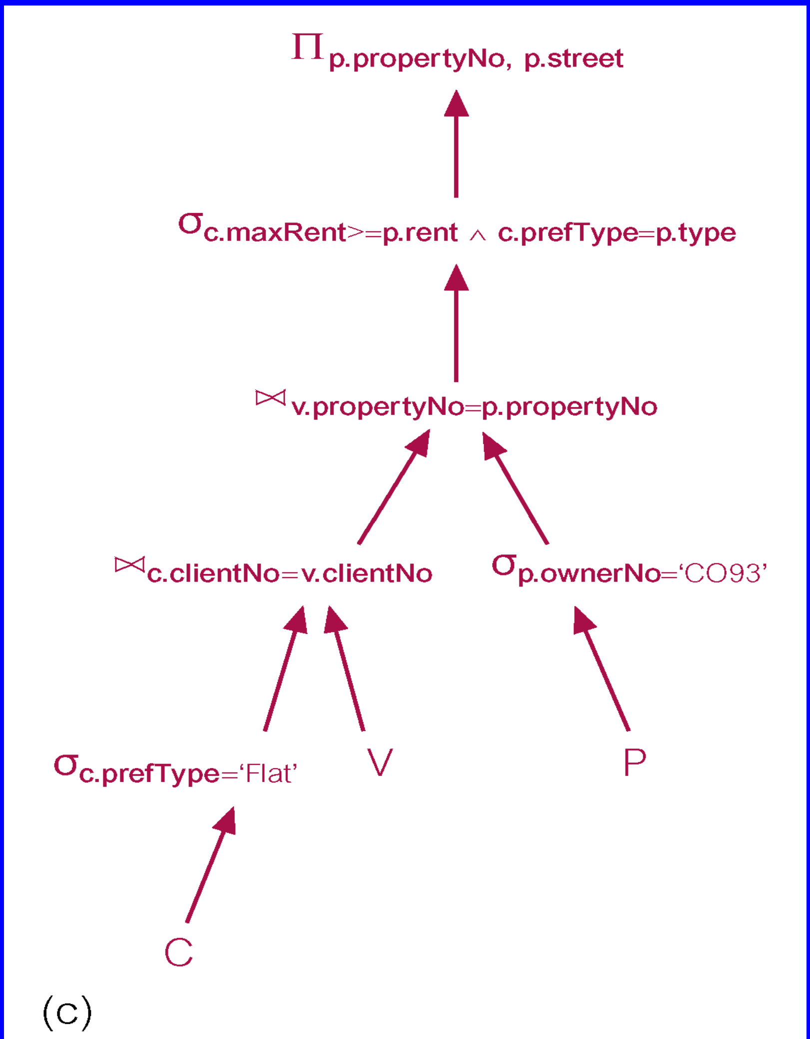
```
SELECT p.propertyNo, p.street
FROM Client c, Viewing v, PropertyForRent p
WHERE   c.prefType = 'Flat' AND
        c.clientNo = v.clientNo AND
        v.propertyNo = p.propertyNo AND
        c.maxRent >= p.rent AND
        c.prefType = p.type AND
        p.ownerNo = 'CO93';
```

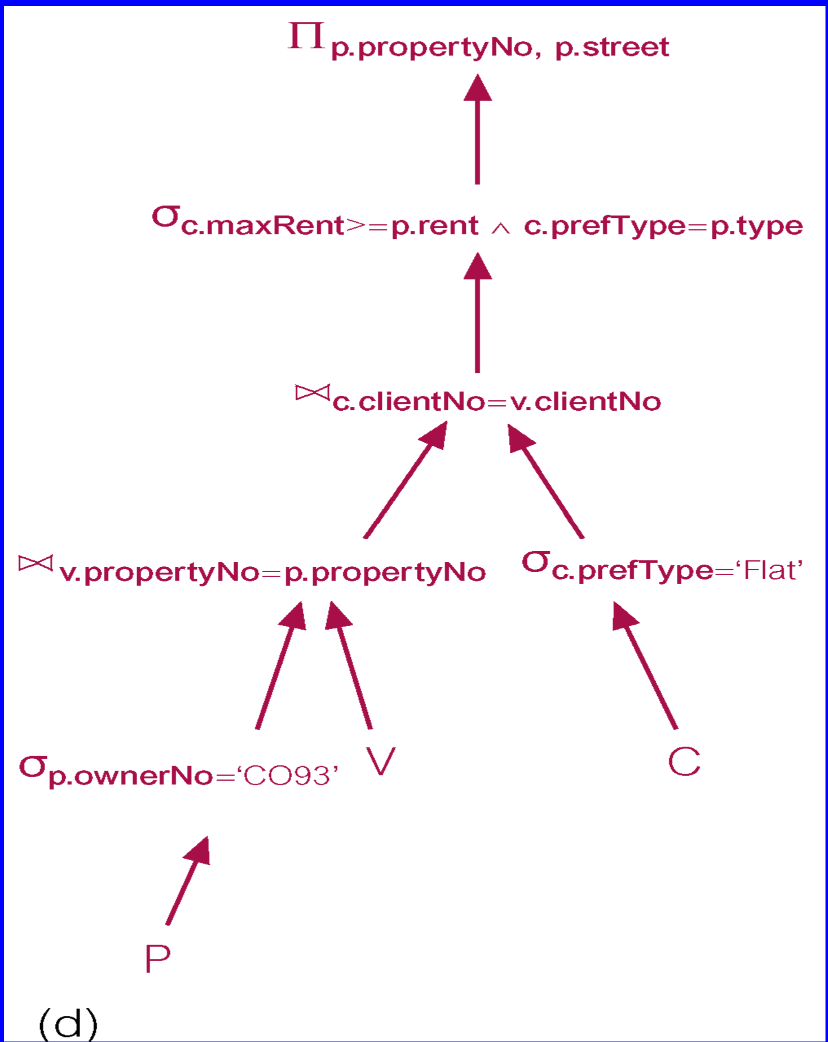
Example 21.3 Use of Transformation Rules



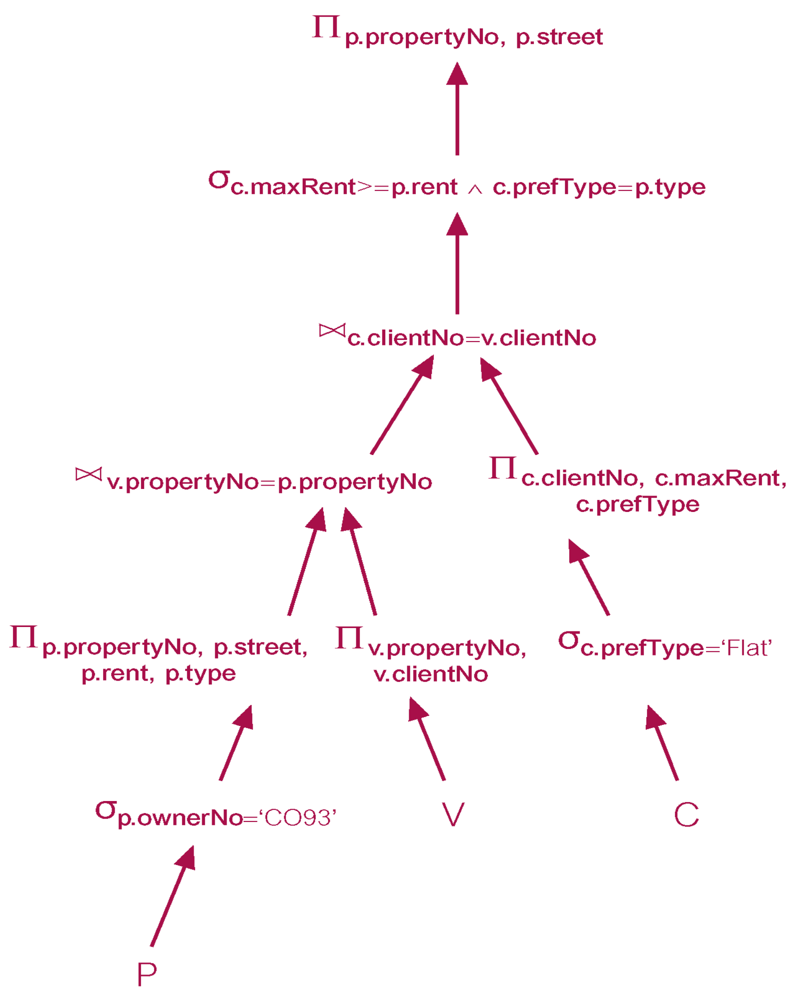


Example 21.3 Use of Transformation Rules

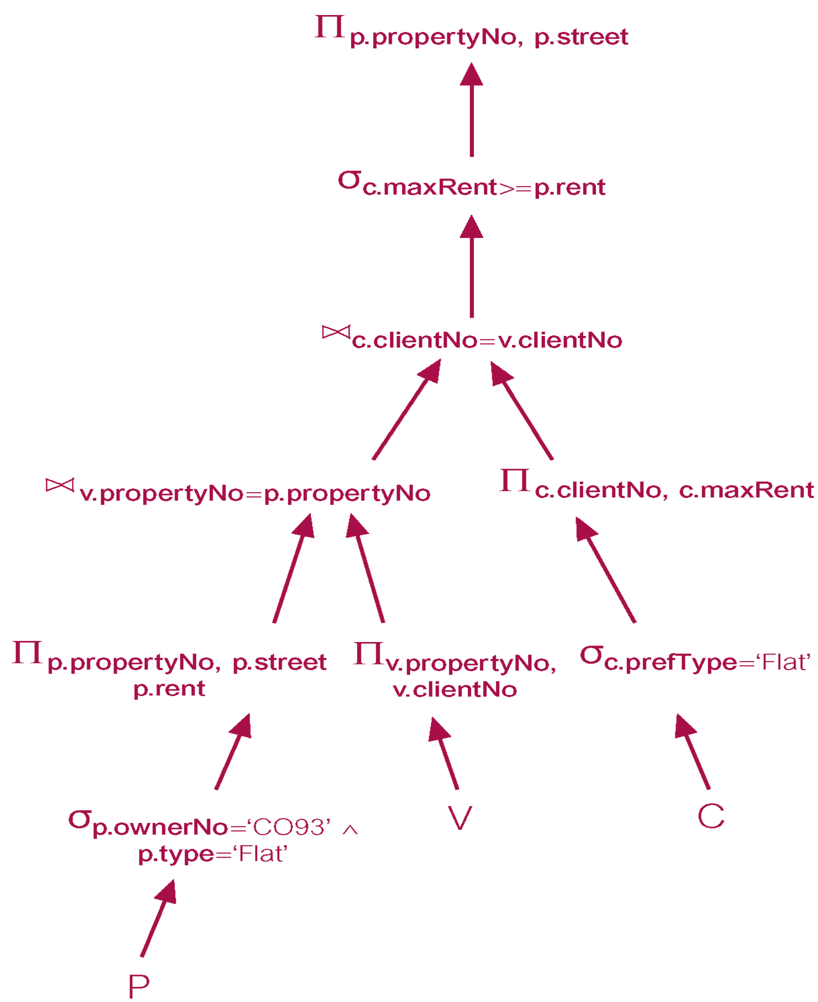




Example 21.3 Use of Transformation Rules



(e)



(f)

Heuristical Processing Strategies

- Perform Selection operations as early as possible.
  - Keep predicates on same relation together.
- Combine Cartesian product with subsequent Selection whose predicate represents join condition into a Join operation.
- Use associativity of binary operations to rearrange leaf nodes so leaf nodes with most restrictive Selection operations executed first.

Heuristical Processing Strategies

- **Perform Projection as early as possible.**
  - **Keep projection attributes on same relation together.**
- **Compute common expressions once.**
  - **If common expression appears more than once, and result not too large, store result and reuse it when required.**
  - **Useful when querying views, as same expression is used to construct view each time.**