Chapter 2

Query Processing and Optimization

Chapter 2 - 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.**
- **Types of database statistics required to estimate cost of operations.**

Chapter 2 – Objectives Cont'd…

- **How pipelining can be used to improve efficiency of queries.**
- **Difference between materialization and pipelining.**

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 a 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.**
- **Practically , both techniques are combined together.**

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

Query Processing (QP)

Defined as: Activities involved in retrieving data from the database.

- involves parsing, validating, optimizing, and executing of a query

- **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

- **Defined as: the Activity of choosing an efficient execution strategy for processing a 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.**
- **Both Approaches of optimizations make use of Database Statistics.**

Sample Relations: Branch and Staff

Attributes

Branch

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

Relation

Cardinality

Degree

Primary key

Foreign key

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

Relation

Example - 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 - Different Strategies

- **Three equivalent RA queries are possible:**

(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 - 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 are stored on disk;**
  - **cost of the final write is ignored(since it is same for all);**
  - ***tuples are accessed one at a time*.**

Analysis of each Query Expression

- ***Let n, m , i, j be all staff, all branches, managers and London branches respectively***
- ***Analysis Q#1:***
  1. *read each tuple from the two relations →n+m reads*
  2. *create a table of the Cartesian product → nXm writes*
  3. *test each tuple of step 2 →nXm read*
- ***Total No. of Disk access: ➔2(nXm) +n+m***
- ***Analysis Q#2:***
  1. *read each tuple from the two relations →n+m reads*
  2. *create a table of the Join → n writes*
  3. *test each tuple of step 2 →n read*
- ***Total No. of Disk access: ➔3(n) +m***
- ***Analysis Q#3:***
  1. *read each tuple from the two relations →n+m reads*
  2. *create a table for Manager staff and London Branches →i+j writes*
  3. *Create a join of Manager Staff and London Branches →i+j reads*
- ***Total No. of Disk access: ➔n+m+ i+j +i+j= (n+m+2*(i+j))***

Example - Cost Comparison

- **Cost (in disk accesses) are:**

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

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

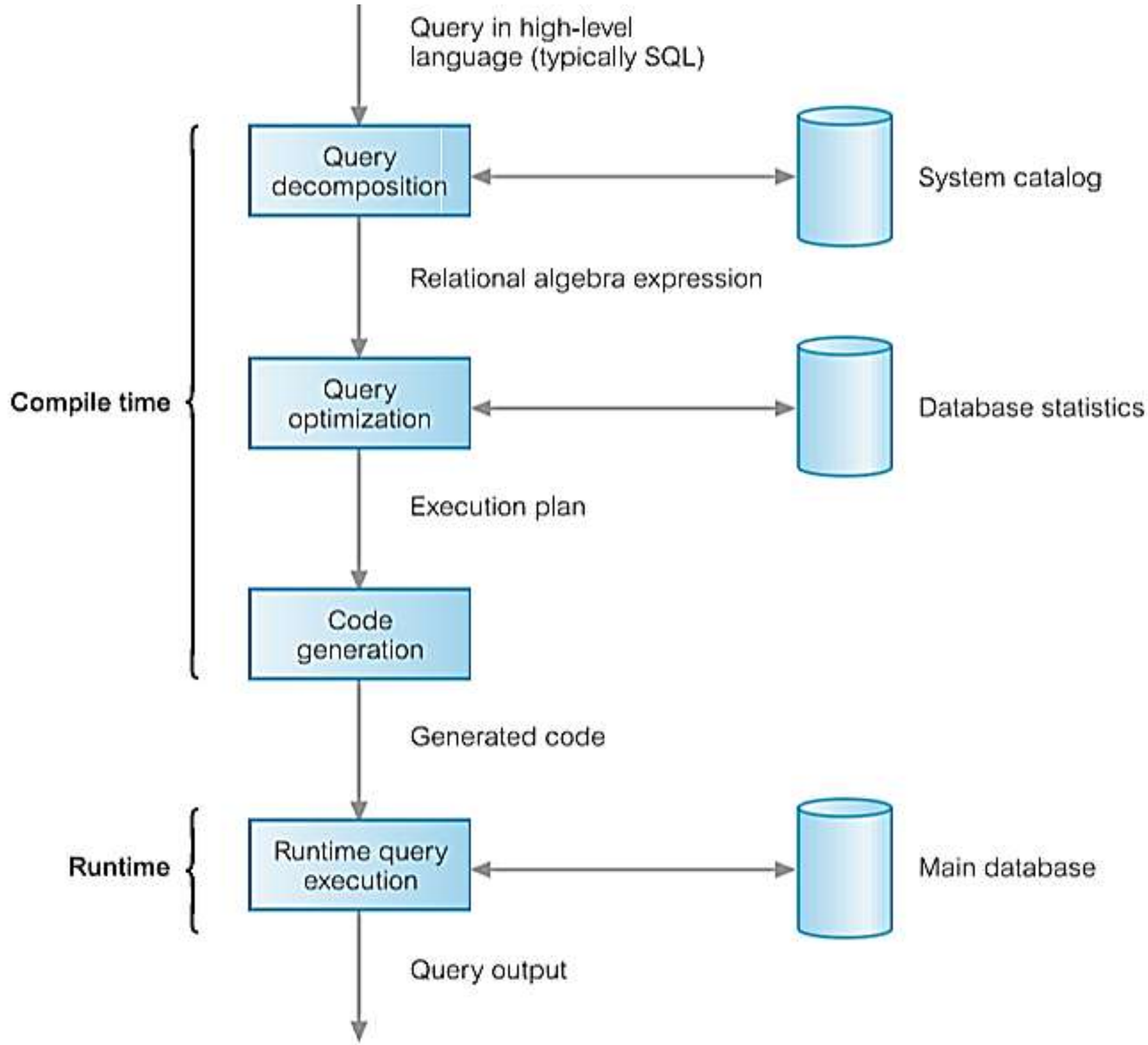
(3) 1000 + 2*50 + 5 + (50 + 5) = 1 160

- **Cartesian product and join operations much more expensive than selection. Hence, the third option significantly reduces size of relations being joined together.**
- **Therefore, the third option is the optimal strategy.**

Phases of Query Processing

- **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

- **The Two times when the 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 re-run.**
- **Could use a hybrid approach to overcome this.**
  - **What kind is the hybrid (how does it work?)?**

Query Decomposition

- **Aims are to transform high-level query into RA query and check that query is syntactically (parsing) and semantically (Validation) correct.**
- **Typical stages are in this phase:**
  - **analysis,**
  - **normalization,**
  - **semantic analysis,**
  - **simplification,**
  - **query restructuring.**

Analysis

- **Analyze query lexically and syntactically using compiler techniques (Against the System catalog).**
  - **Verify relations and attributes exist.**
  - **Verify operations are appropriate for object type.**

Analysis - Example

SELECT staff_no

FROM Staff

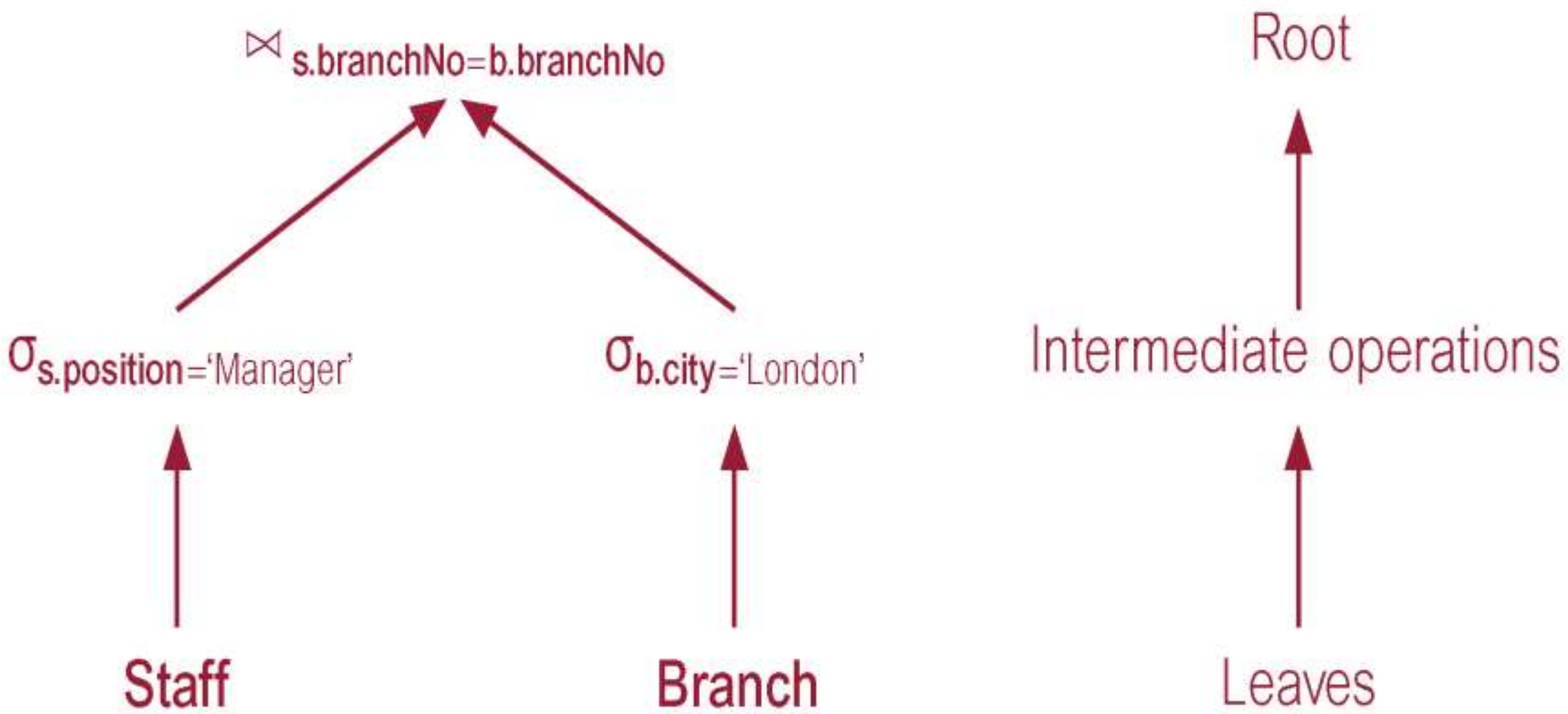
WHERE position > 10;

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

Analysis

- **After Lexical and Syntactical analysis, query is transformed into some internal representation, more suitable for processing.**
- **Some kind of query tree is typically chosen and 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 (of operations) is directed from leaves to root and from left to right.**

Example - R.A.T.



⋈ s.branchNo=b.branchNo
σ s.position='Manager'
σ b.city='London'
Staff
Branch
Root
Intermediate operations
Leaves

Normalization

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

<u>Conjunctive normal form:</u>

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

<u>Disjunctive normal form:</u>

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

Semantic Analysis

- **Rejects normalized queries that are** ***incorrectly formulated or contradictory.***
  - **Query is <u>incorrectly formulated</u> if components do not contribute to the generation of result, which may happen if some join specifications are missing.**
  - **Query is <u>contradictory</u> if its predicate cannot be satisfied by any tuple.**
- **Algorithms to determine correctness exist only for the subset of queries that do not contain disjunction and negation.**

Semantic Analysis

- **For these queries, could construct:**
  - **A relation connection graph.**
  - **Normalized attribute connection graph.**

Relation connection graph

Create node for each relation **and** ***node for result*****. Create edges between two nodes that represent a join, and edges between nodes hat represent the source of Projection operations.**

- **If *not connected*, query is *incorrectly formulated*.**

Semantic Analysis Cont'd…

Attribute Connection Graph

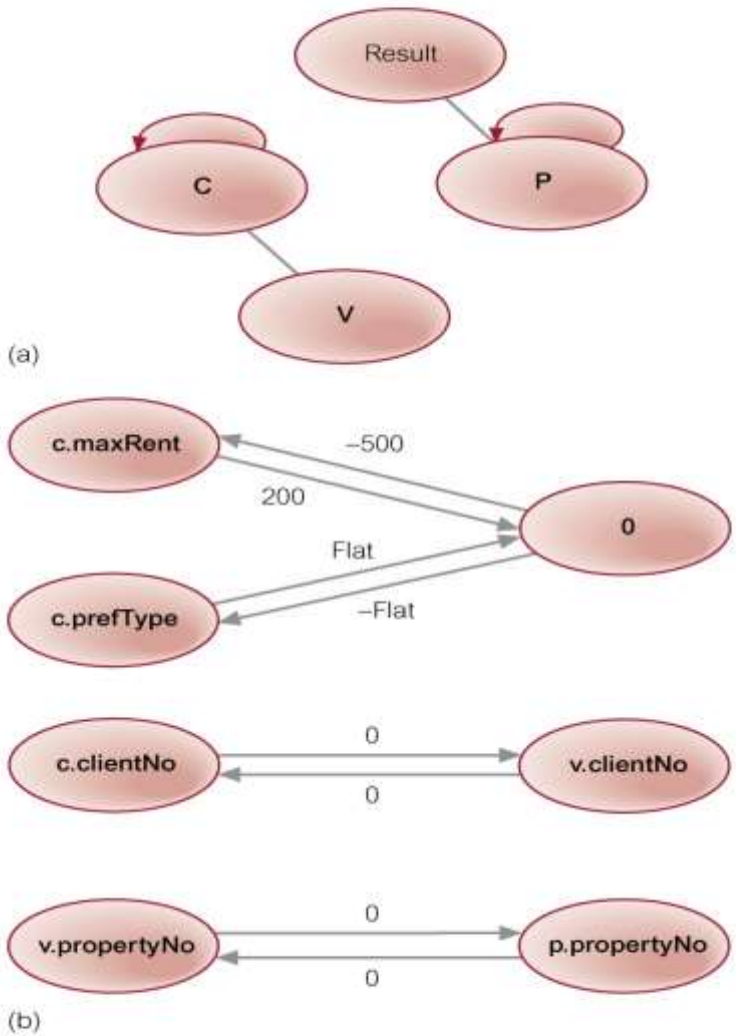
- If the graph has a cycle for which the valuation sum is negative, the *query is contradictory*.
- To construct a normalized attribute connection graph, we create a node for each reference to an attribute, or constant 0.
- We then create a directed edge between nodes that represent a join, and a directed edge between an attribute node and a constant 0 node that represents a selection operation.
- Next, we weight the edges $a \rightarrow b$ with the value c, if it represents the inequality condition ($a \leq b + c$), and weight the edges $0 \rightarrow a$ with the value $-c$, if it represents the inequality condition ($a \geq c$).

Example - Checking Semantic Correctness

SELECT p.propertyNo, p.street
FROM Client c, Viewing v, PropertyForRent p
WHERE c.clientNo = v.clientNo AND
c.maxRent >= 500 AND
c.prefType = 'Flat' AND p.ownerNo = 'CO93';

- **Relation connection graph not fully connected, so query is not correctly formulated.**
- ***Have omitted the join condition* (v.propertyNo = p.propertyNo) .**

Example - Checking Semantic Correctness



Relation Connection graph

Normalized attribute connection graph

Example - Checking Semantic Correctness

```
SELECT p.propertyNo, p.street
FROM Client c, Viewing v, PropertyForRent p
WHERE  c.maxRent > 500 AND
       c.clientNo = v.clientNo AND
       v.propertyNo = p.propertyNo AND
       c.prefType = 'Flat' AND c.maxRent < 200;
```

- **Normalized attribute connection graph has cycle between nodes c.maxRent and 0 with negative valuation sum, so query is contradictory.**

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 for such simplifications.
- Assuming user has appropriate access privileges, first apply well-known idempotency rules of Boolean algebra.
- Examples: for two predicates p and q , P V~P= True, p $\wedge$ (qv~q)=P …etc.
- In Relational Algebra, we have Transformation rules to do so…

Transformation Rules for RA Operations

- The Heuristic Approach to QO is based on the transformation rules of Relational Algebra.
- 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.

$$\Pi_L \Pi_M \ldots \Pi_N(R) = \Pi_L (R)$$ **, provided that L is in M and M is in N**

- **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 ofTheta 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:**

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

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

  where $p \in \{A_1, A_2, \ldots, A_n\}$ which are the attributes of R.

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:**

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

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

Transformation Rules for RA Operations

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

- **If projection list is of the 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, then Projection and Theta join commute as:**

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

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).

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

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

$$R - S \neq S - R$$

Transformation Rules for RA Operations

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

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

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

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

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 ofTheta 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:**

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

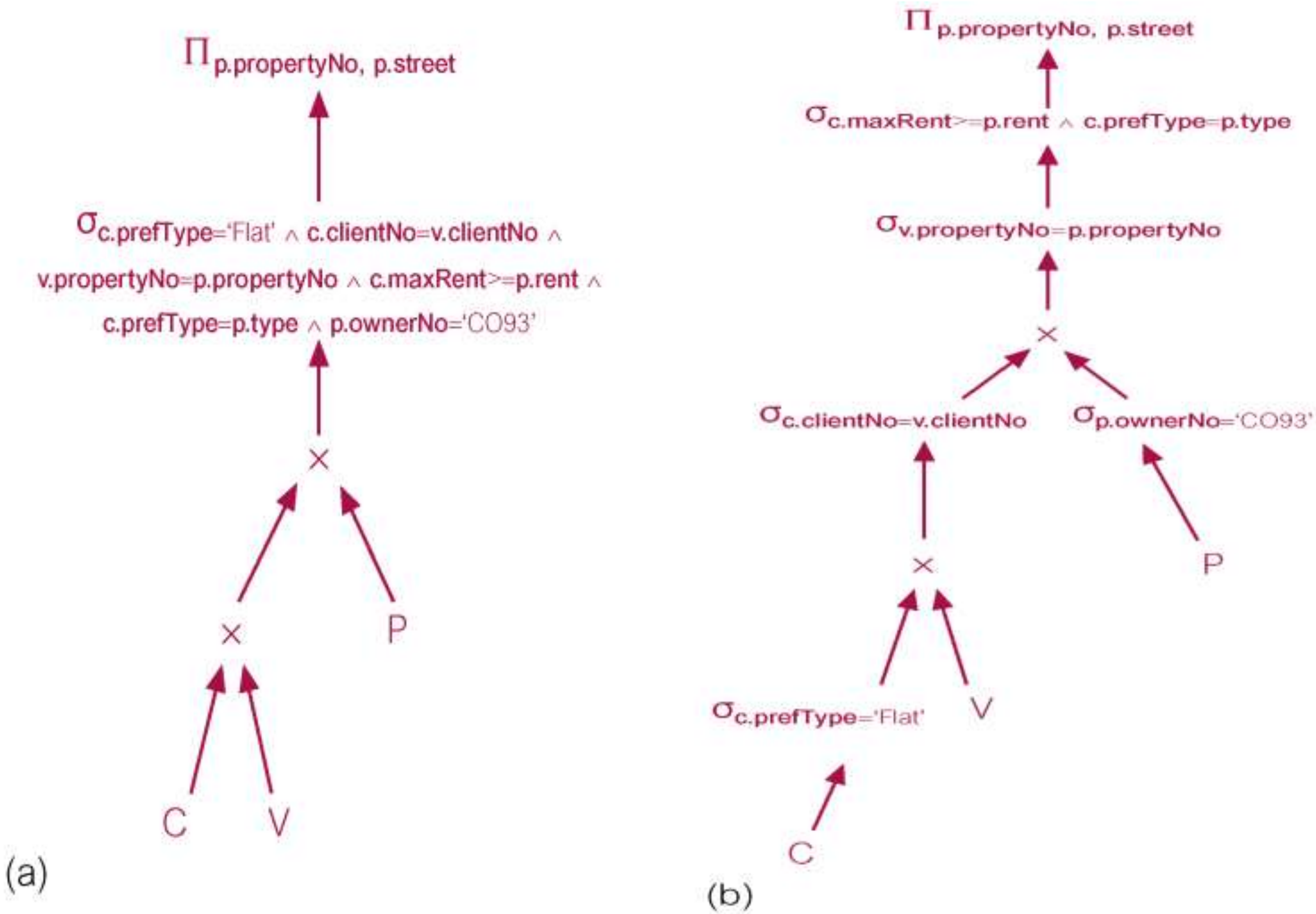
$$\text{Staff} \bowtie_{\text{staff.staffNo=PropertyForRent.staffNo} \wedge \text{staff.lName=Owner.lName}} (\text{PropertyForRent} \bowtie_{\text{ownerNo}} \text{Owner})$$

Example Use of Transformation Rules

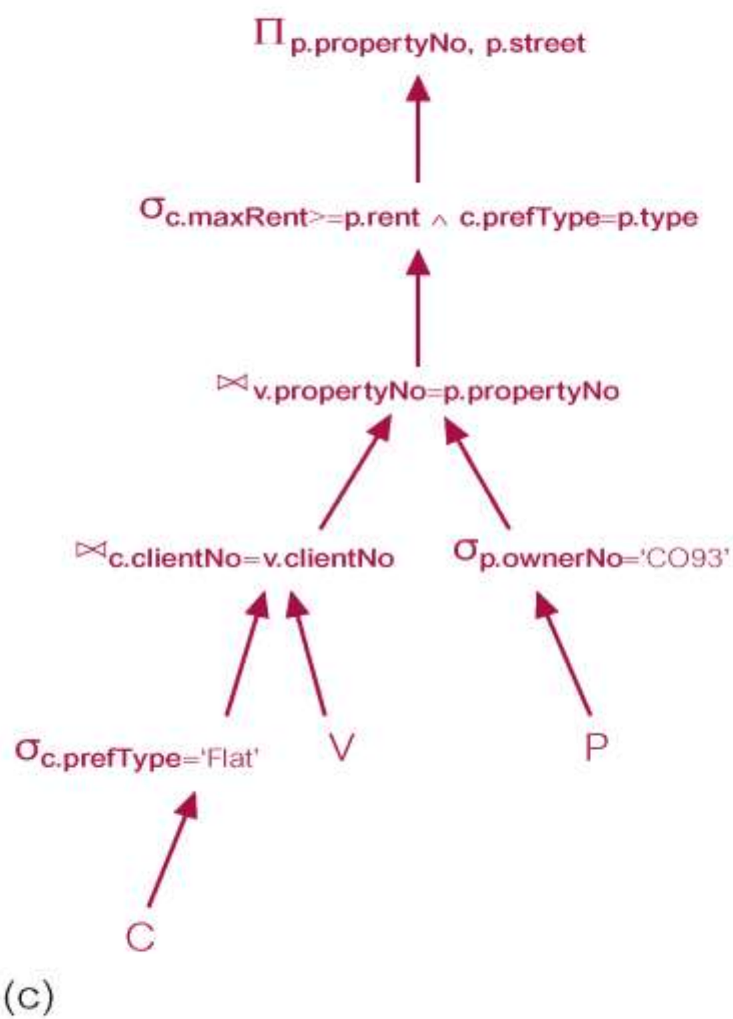
For prospective renters of flats, find properties that match their requirements and owned by CO93.

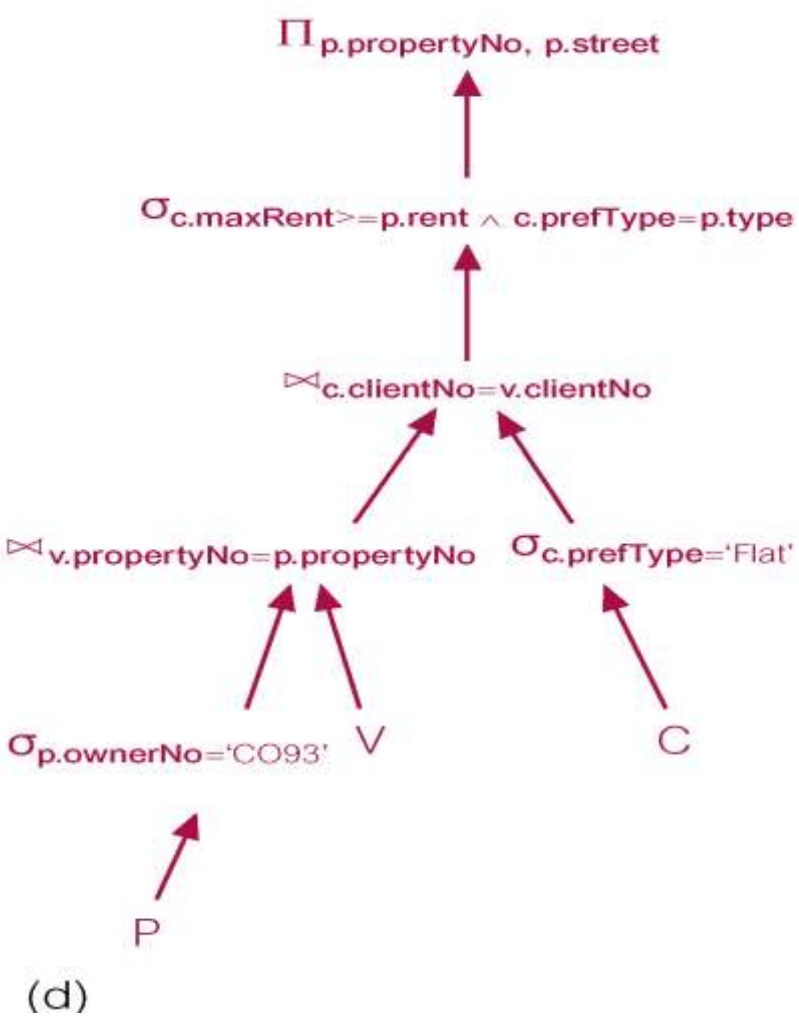
```
SELECT p.propertyNo, p.street
FROM Client c, Viewing v, PropertyForRent p
WHERE   c.prefType = 'Flat' AND
        c.clientNo = v.clientNo AND
        v.propertyNo = p.propertyNo AND
        c.maxRent >= p.rent AND
        c.prefType = p.type AND
        p.ownerNo = 'CO93';
```

Example Use of Transformation Rules

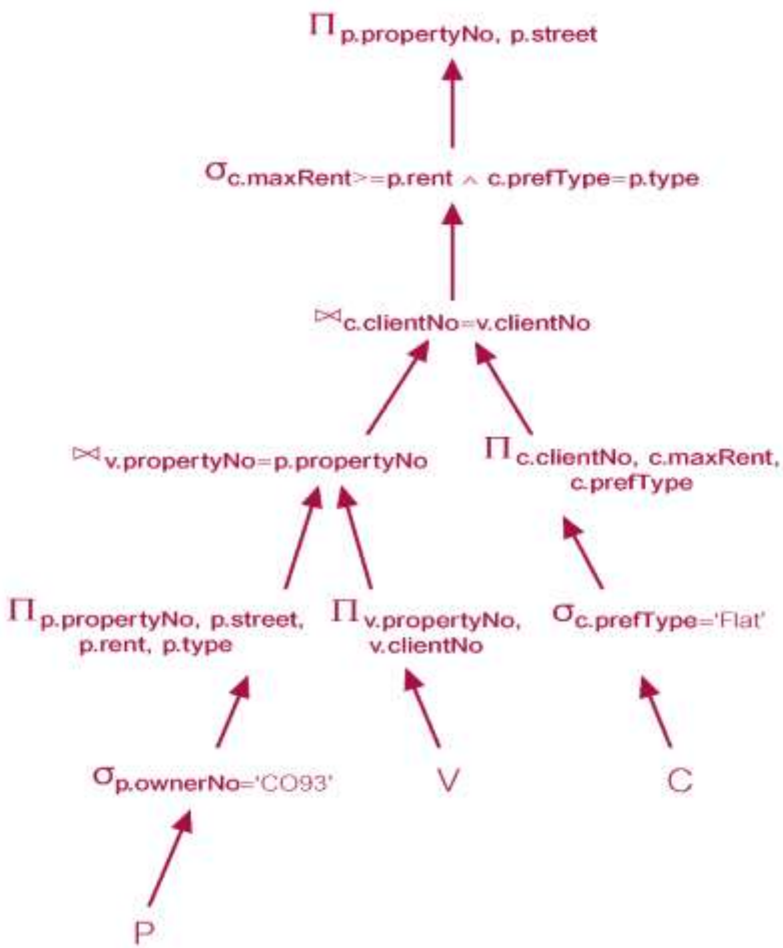


Example Use of Transformation Rules

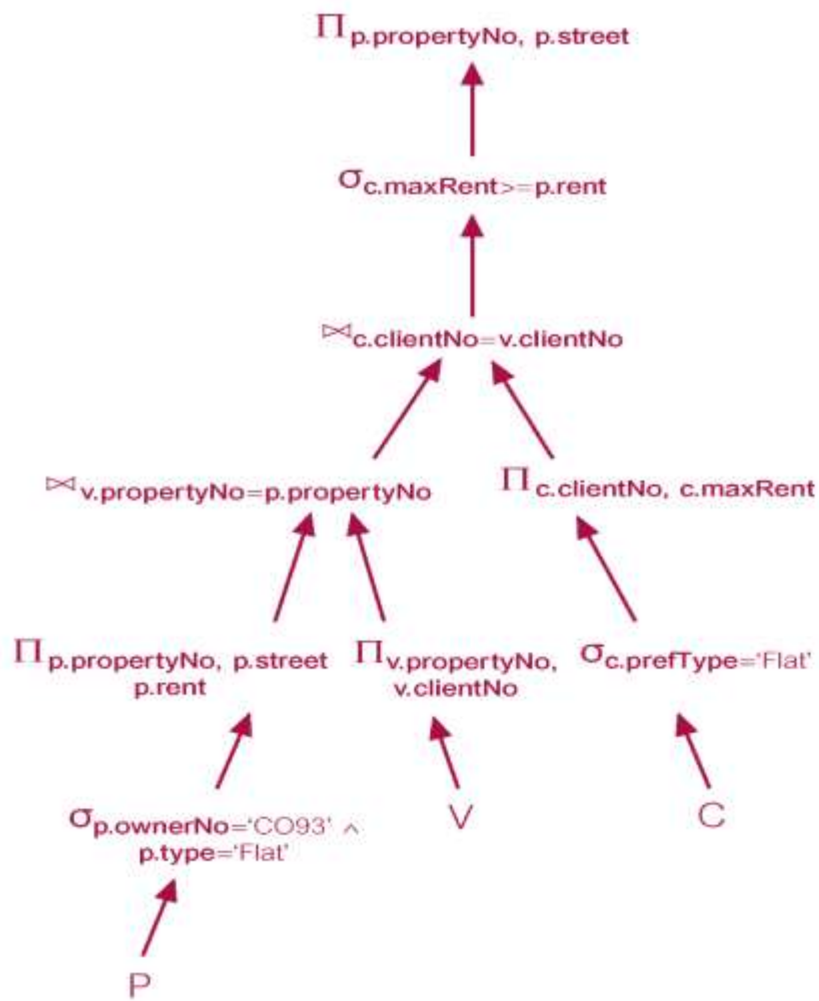




Example Use of Transformation Rules



(e)



(f)

Heuristical Processing Strategies

1. **Perform Selection operations as early as possible.**
   - **Keep predicates on same relation together. Conjunctive selects →individual selects(cascade of selection)**
   - **Push Selections to the respective tables**
2. **Use associativity of binary operations to rearrange leaf nodes so leaf nodes with most restrictive Selection operations executed first.(reduces size of join)**
3. **Combine *Cartesian product with subsequent selection* whose predicate represents join condition into a Join operation.**

Heuristic Processing Strategies…

4. **Perform Projection as early as possible.**
   - **Keep projection attributes on same relation together.**
   - **Push Projection to the respective tables.**
5. **Compute common expressions once (prefType in the example seen)**

In Each step, consider the Relational Algebra transformation rules

Cost Estimation for RA Operations

- **Many different ways of implementing RA operations.**
- **Aim of QO is to choose most efficient one.**
- **Use formulae that estimate costs for a number of options, and select one with lowest cost.**
- **Consider only cost of disk access, which is usually dominant cost in QP.**
- **Many estimates are based on cardinality of the relation, so need to be able to estimate this.**
- **Cost Estimation is then done for the operations involved in an expression.**
- **The Expression with the lowest cost is chosen for execution.**

Database Statistics

- Success of estimation depends on amount and currency of statistical information that DBMS holds.
- Keeping statistics current can be problematic.
- If statistics updated every time tuple is changed, this would impact performance.
- DBMS could update statistics on a periodic basis, for example nightly, or whenever the system is idle.
  - eg. MSSQL Server has Maintenance Workflow for updating statistics
- Another approach taken by some systems is to make it the users' responsibility to indicate that the statistics should be updated

Typical Statistics for Relation R

nTuples(R) - number of tuples in R.

bFactor(R) - blocking factor of R (Number of tuples in a block).

nBlocks(R) - number of blocks required to store R:

nBlocks(R) = [nTuples(R)/bFactor(R)]

Typical Statistics for Attribute A of Relation R

$nDistinct_A(R)$ - number of distinct values that appear for attribute A in R.

$min_A(R), max_A(R)$

- **minimum and maximum possible values for attribute A in R.**

$SC_A(R)$ - *selection cardinality* of attribute A in R.

Average number of tuples that satisfy an equality condition on attribute A.

Pipelining

- **<u>Materialization</u> - output of one operation is stored in temporary relation for processing by next. (heuristic approach)**
- **Could also pipeline results of one operation to another without creating temporary relation.**
- **Known as <u>pipelining</u> or <u>on-the-fly</u> processing or stream-based processing(in-memory stream data)**
- **Pipelining can save on cost of creating temporary relations and reading results back in again.**
- **Generally, pipeline is implemented as separate process or thread.**

Assignment- Due next week same time.

- Assume the following Tables
  - **Employee** (EID, Fn, Ln, DOB,Position, Gender, Salary,DOE)
  - **Project** (ProjectID, PName,StartDate, EndDate, PBudget)
  - **Works_ON**(EID,ProjectId,DateStart,DateEnd, Bonus)

Questions: 1) Write the SQL for the following query

2) Translate the SQL in Question 1 to an initial Relational Algebra Expression (RAE).

3)Optimize the RAE in Question 2. (provide both RA **Expressions** and the corresponding **Query Tree**)

Query: Find the Names, salary and Bonuses of Employees that were born before January 1, 1970 and have worked on a project named "GERD".

Note: Assignment **should be completed in groups consisting of only two students per group**.