Database System Concepts

Chapter 14: Query Optimization

Departamento de Engenharia Informática
Instituto Superior Técnico

1st Semester
2009/2010

Outline

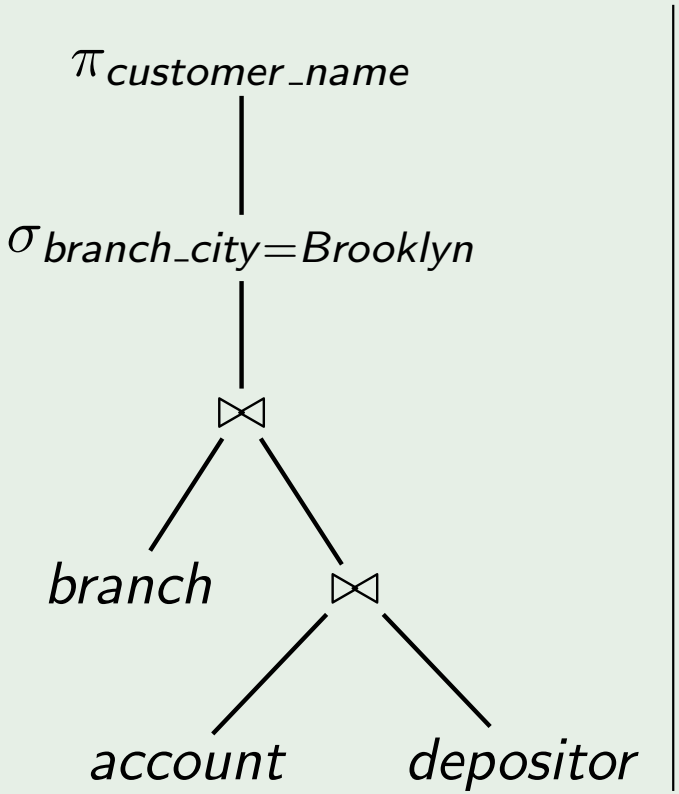
Outline

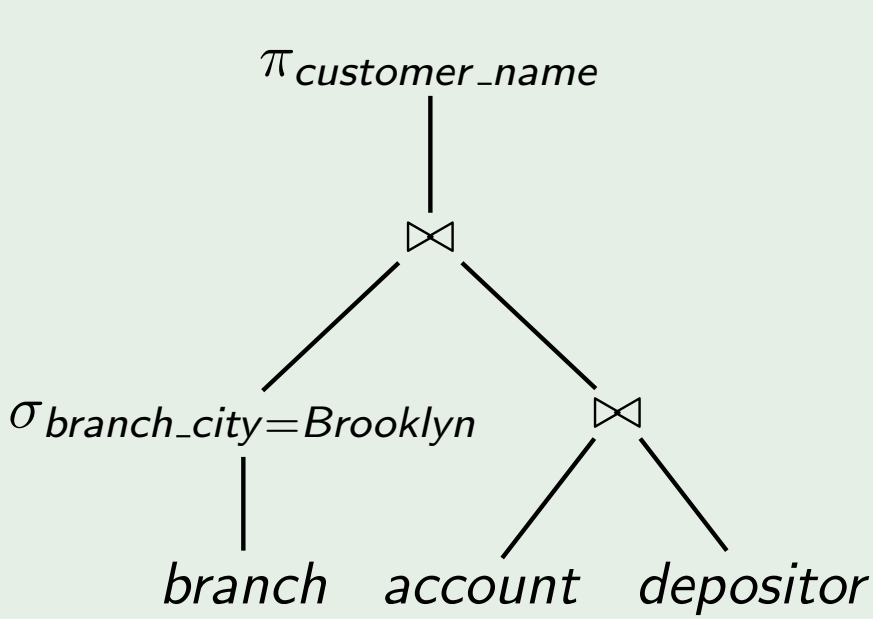
Query Evaluation

- Alternative ways of evaluating a given query
  - Equivalent expressions
  - Different algorithms for each operation
- Cost difference between a good and a bad way of evaluating a query can be enormous
- Need to estimate the cost of operations
  - Statistical information about relations. Examples:
    - number of tuples
    - number of distinct values for an attributes
    - ...
  - Statistics estimation for intermediate results
    - to compute cost of complex expressions

Equivalent Expressions

- Relations generated by two equivalent expressions have the same set of attributes and contain the same set of tuples
  - although their tuples/attributes may be ordered differently





Cost-Based Optimization

- Generation of query-evaluation plans for an expression involves several steps:
  - Generating logically equivalent expressions using equivalence rules
  - Annotating resultant expressions to get alternative query plans
  - Choosing the cheapest plan based on estimated cost
- The overall process is called cost-based optimization

Outline

Transformation of Relational Expressions

- Two relational algebra expressions are said to be equivalent if on every legal database instance the two expressions generate the same set of tuples
  - Note: order of tuples is irrelevant
- 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 on every legal database instance the two expressions generate the same multiset of tuples
- An equivalence rule says that expressions of two forms are equivalent
  - Can replace expression of first form by second, or vice versa

Equivalence Rules I

1) Conjunctive selection operations can be deconstructed into a sequence of individual selections

$$\sigma_{\theta_1 \wedge \theta_2}(E) = \sigma_{\theta_1}(\sigma_{\theta_2}(E))$$

2) Selection operations are commutative

$$\sigma_{\theta_1}(\sigma_{\theta_2}(E)) = \sigma_{\theta_2}(\sigma_{\theta_1}(E))$$

3) Only the last in a sequence of projection operations is needed, the others can be omitted

$$\pi_{L_1}(\pi_{L_2}(\ldots(\pi_{L_n}(E))\ldots)) = \pi_{L_1}(E)$$

Equivalence Rules II

4) Selections can be combined with Cartesian products and theta joins

$$\sigma_\theta(E_1 \times E_2) = E_1 \bowtie_\theta E_2$$
$$\sigma_{\theta_1}(E_1 \bowtie_{\theta_2} E_2) = E_1 \bowtie_{\theta_1 \wedge \theta_2} E_2$$

5) Theta-join operations and natural joins are commutative

$$E_1 \bowtie_\theta E_2 = E_2 \bowtie_\theta E_1$$

Equivalence Rules III

6) Join operations are associative

(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

$$(E_1 \bowtie_{\theta_1} E_2) \bowtie_{\theta_2 \wedge \theta_3} E_3 = E_1 \bowtie_{\theta_1 \wedge \theta_3} (E_2 \bowtie_{\theta_2} E_3)$$

where θ_2 involves attributes from only E_2 and E_3

Equivalence Rules IV

Example

$$(customer \bowtie_{c.c\_name=d.c\_name} depositor) \bowtie_{d.a\_number=a.a\_number \wedge balance>1000} account$$

$$=$$

$$customer \bowtie_{c.c\_name=d.c\_name \wedge balance>1000} (depositor \bowtie_{d.a\_number=a.a\_number} account)$$

Equivalence Rules V

7) The selection operation distributes over the theta join operation under the following two conditions:

(a) When all the attributes in θ_0 involve only the attributes of one of the expressions (E_1) being joined

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

(b) When θ_1 involves only the attributes of E_1 and θ_2 involves only the attributes of E_2

$$\sigma_{\theta_1 \wedge \theta_2}(E_1 \bowtie_\theta E_2) = (\sigma_{\theta_1}(E_1)) \bowtie_\theta (\sigma_{\theta_2}(E_2))$$

Equivalence Rules VI

Example

$$\sigma_{c\_name=Smith \wedge balance>1000}(depositor \bowtie account)$$

$$=$$

$$\sigma_{c\_name=Smith}(depositor) \bowtie \sigma_{balance>1000}(account)$$

Equivalence Rules VII

8) The projections operation distributes over the theta join operation as follows:

(a) Let L_1 and L_2 be attributes from E_1 and E_2. If the join involves only attributes in $L_1 \cup L_2$

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

(b) Let L_3 be attributes of E_1 that are involved in the join condition, but are not in $L_1 \cup L_2$, and let L_4 be attributes of E_2 that are involved in the join condition, but are not in $L_1 \cup L_2$

$$\pi_{L_1 \cup L_2}(E_1 \bowtie_\theta E_2) = \pi_{L_1 \cup L_2}((\pi_{L_1 \cup L_3}(E_1)) \bowtie_\theta (\pi_{L_2 \cup L_4}(E_2)))$$

Equivalence Rules VIII

Example

$$\pi_{b\_name,c\_name}(account \bowtie_{a.a\_number=d.a\_number} depositor)$$

$$=$$

$$\pi_{b\_name,c\_name}((\pi_{b\_name,a\_number}(E_1)) \bowtie_{a.a\_number=d.a\_number} (\pi_{c\_name,a\_number}(E_2)))$$

Equivalence Rules IX

9) The set operations union and intersection are commutative

$$E_1 \cup E_2 = E_2 \cup E_1$$
$$E_1 \cap E_2 = E_2 \cap E_1$$

10) Set union and intersection are associative

$$(E_1 \cup E_2) \cup E_3 = E_1 \cup (E_2 \cup E_3)$$
$$(E_1 \cap E_2) \cap E_3 = E_1 \cap (E_2 \cap E_3)$$

Equivalence Rules X

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 $\cap$. Also

$$\sigma_\theta(E_1 - E_2) = \sigma_\theta(E_1) - E_2$$

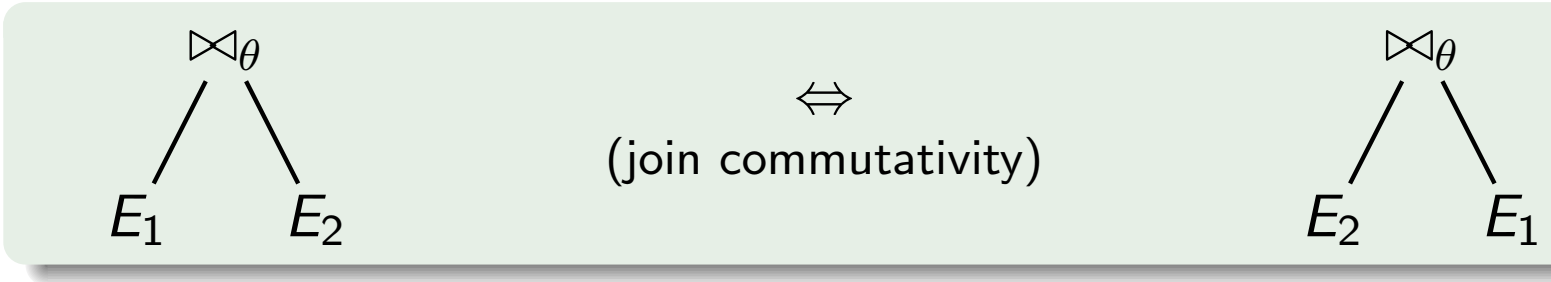
and similarly for $\cup$ and $\cap$

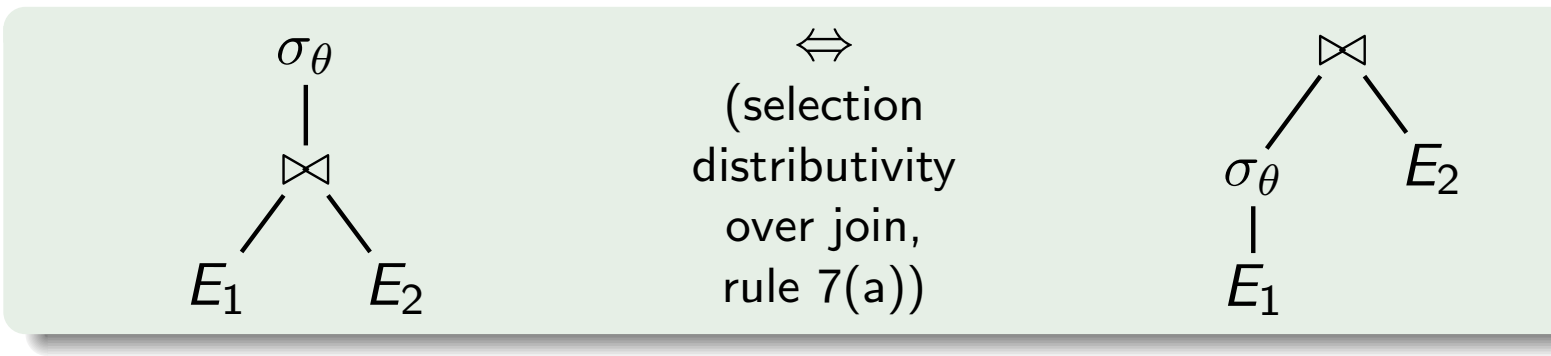
12) The projection operation distributes over union

$$\pi_L(E_1 \cup E_2) = \pi_L(E_1) \cup \pi_L(E_2)$$

Other rules also exist for outer joins, aggregations, ...

Graphical Examples





Transformation Example

- Query: Find the names of all customers who have an account at some branch located in Brooklyn

$$\pi_{customer\_name}(\sigma_{branch\_city=Brooklyn}(branch \bowtie (account \bowtie depositor)))$$

- Transformation using rule 7(a)

$$\pi_{customer\_name}((\sigma_{branch\_city=Brooklyn}(branch)) \bowtie (account \bowtie depositor))$$

- Performing the selection as early as possible reduces the size of the relation to be joined

Multiple Transformations

- Query: Find the names of all customers with an account at a Brooklyn branch whose account balance is over $1000

$$\pi_{customer\_name}((\sigma_{branch\_city=Brooklyn \wedge balance>1000} (branch \bowtie (account \bowtie depositor)))$$

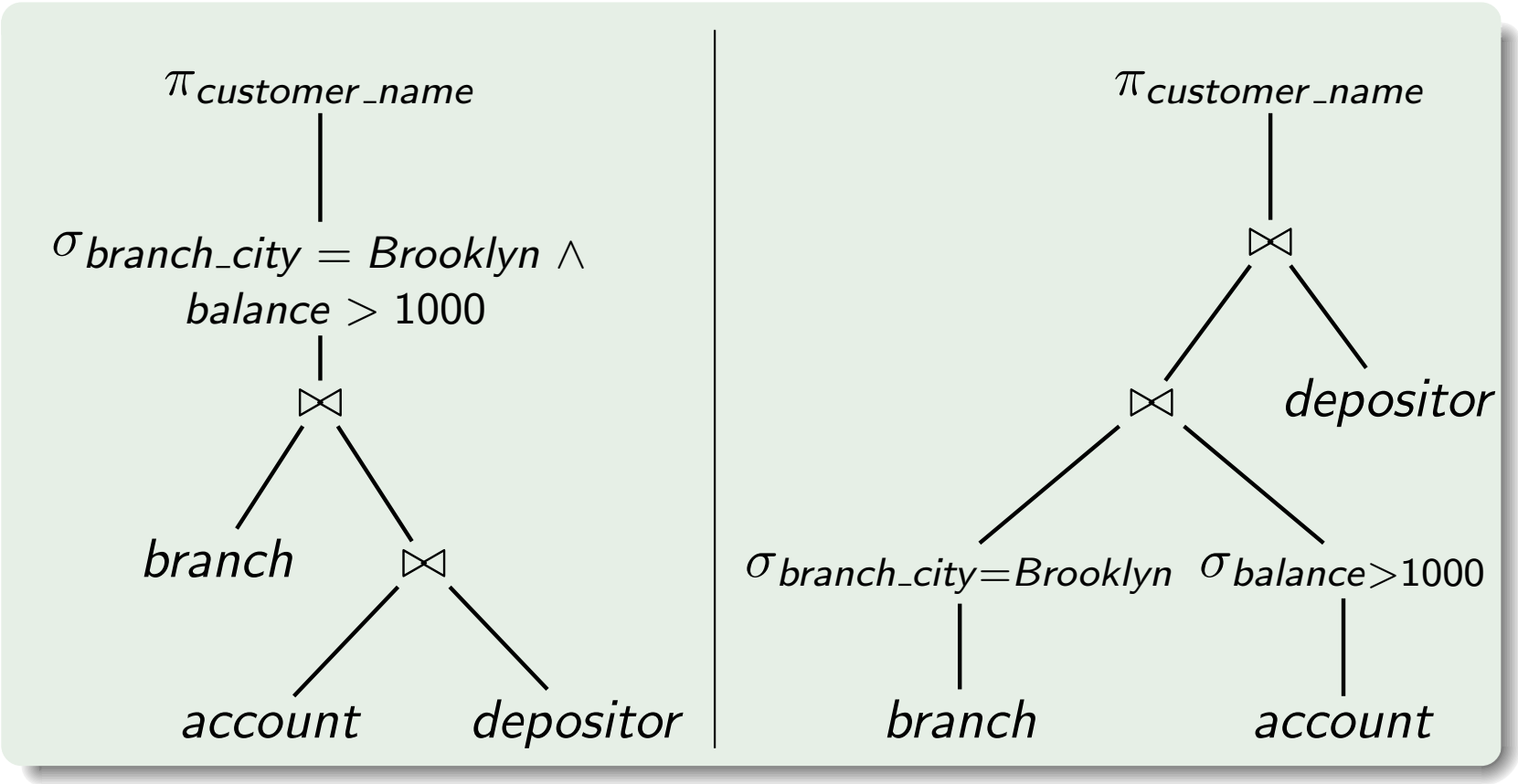
- Transformation using join associatively (rule 6(a)):

$$\pi_{customer\_name}((\sigma_{branch\_city=Brooklyn \wedge balance>1000} (branch \bowtie account) \bowtie depositor))$$

- Second form provides an opportunity to apply the "perform selections early" rule, resulting in the expression

$$\pi_{customer\_name}((\sigma_{branch\_city=Brooklyn}(branch) \bowtie \sigma_{balance>1000}(account)) \bowtie depositor)$$

Multiple Transformations (cont.)



Projection Operation Example

- Consider the query

$$\pi_{customer\_name}((\sigma_{branch\_city=Brooklyn}(branch) \bowtie account) \bowtie depositor)$$

- When we compute $\sigma_{branch\_city=Brooklyn}(branch) \bowtie account$, we get a relation whose schema is $(branch\_name, branch\_city, assets, account\_number, balance)$
- Push projections using equivalence rules 8(a) and 8(b) eliminate unneeded attributes from intermediate results

$$\pi_{customer\_name}(\pi_{account\_number}(\sigma_{branch\_city=Brooklyn}(branch) \bowtie account) \bowtie depositor)$$

Join Ordering Example

- Consider the expression

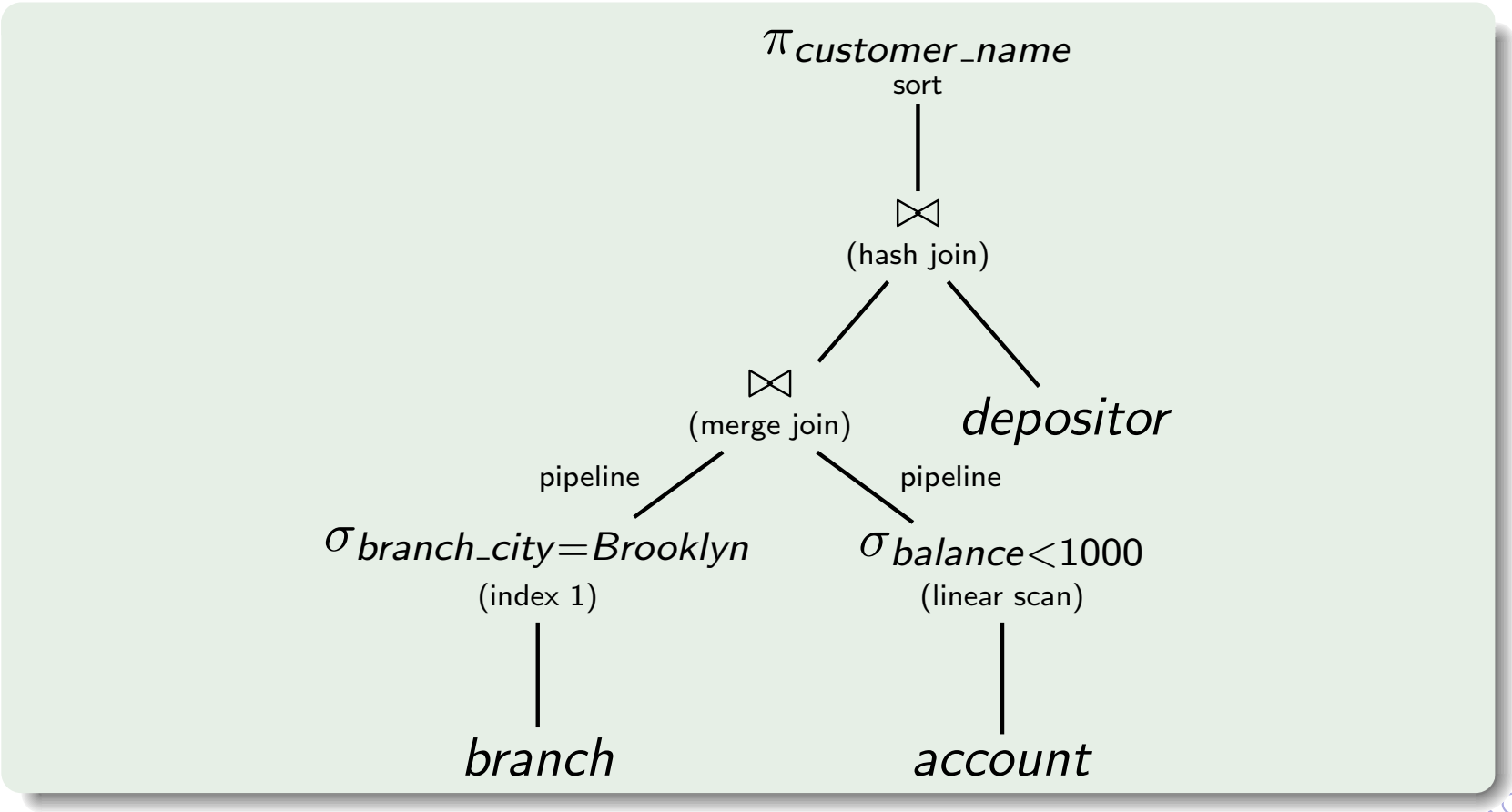
$$\pi_{customer\_name}((\sigma_{branch\_city=Brooklyn}(branch)) \bowtie (account \bowtie depositor))$$

- Could compute $(account \bowtie depositor)$ first, and join result with $\sigma_{branch\_city=Brooklyn}(branch)$
- However, $(account \bowtie depositor)$ is likely to be a large relation...
- ... and only a small fraction of the bank's customers are likely to have accounts in branches located in Brooklyn
- Thus, it is better to compute first

$$\sigma_{branch\_city=Brooklyn}(branch) \bowtie account$$

Evaluation Plan

- An evaluation plan defines exactly what algorithm is used for each operation, and how the execution of the operations is coordinated



Choice of Evaluation Plans

- Must consider the interaction of evaluation techniques when choosing evaluation plans: choosing the cheapest algorithm for each operation independently may not yield best overall algorithm
  - merge-join may be costlier than hash-join, but may provide a sorted output which reduces the cost for an outer level aggregation
  - nested-loop join may provide opportunity for pipelining
- Practical query optimizers incorporate elements of the following two broad approaches:
  1. Search all the plans and choose the best plan in a cost-based fashion
  2. Uses heuristics to choose a plan

Heuristic Optimization

- Cost-based optimization is expensive, even with dynamic programming
- Systems may use heuristics to reduce the number of choices that must be made in a cost-based fashion
- Heuristic optimization transforms the query-tree by using a set of rules that typically (but not in all cases) improve execution performance:
  - Perform selection early (reduces the number of tuples)
  - Perform projection early (reduces the number of attributes)
  - Perform most restrictive selection and join operations before other similar operations
  - Some systems use only heuristics, others combine heuristics with partial cost-based optimization

Typical Heuristic Optimization

1. Deconstruct conjunctive selections into a sequence of single selection operations (rule 1)
2. Move selection operations down the query tree for the earliest possible execution (rules 2, 7(a), 7(b), and 11)
3. Execute first those selection and join operations that will produce the smallest relations (rule 6)
4. Replace Cartesian product operations that are followed by a selection condition by join operations (rule 4(a))
5. Deconstruct and move as far down the tree as possible lists of projection attributes, creating new projections where needed (rules 3, 8(a), 8(b), and 12)
6. Identify those subtrees whose operations can be pipelined, and execute them using pipelining

End of Chapter 14