



Chapter 7(contd.): Query Optimization

Database System Concepts 5th Ed.

©Silberschatz, Korth and Sudarshan

See www.db-book.com for conditions on re-use





Chapter 14: Query Optimization

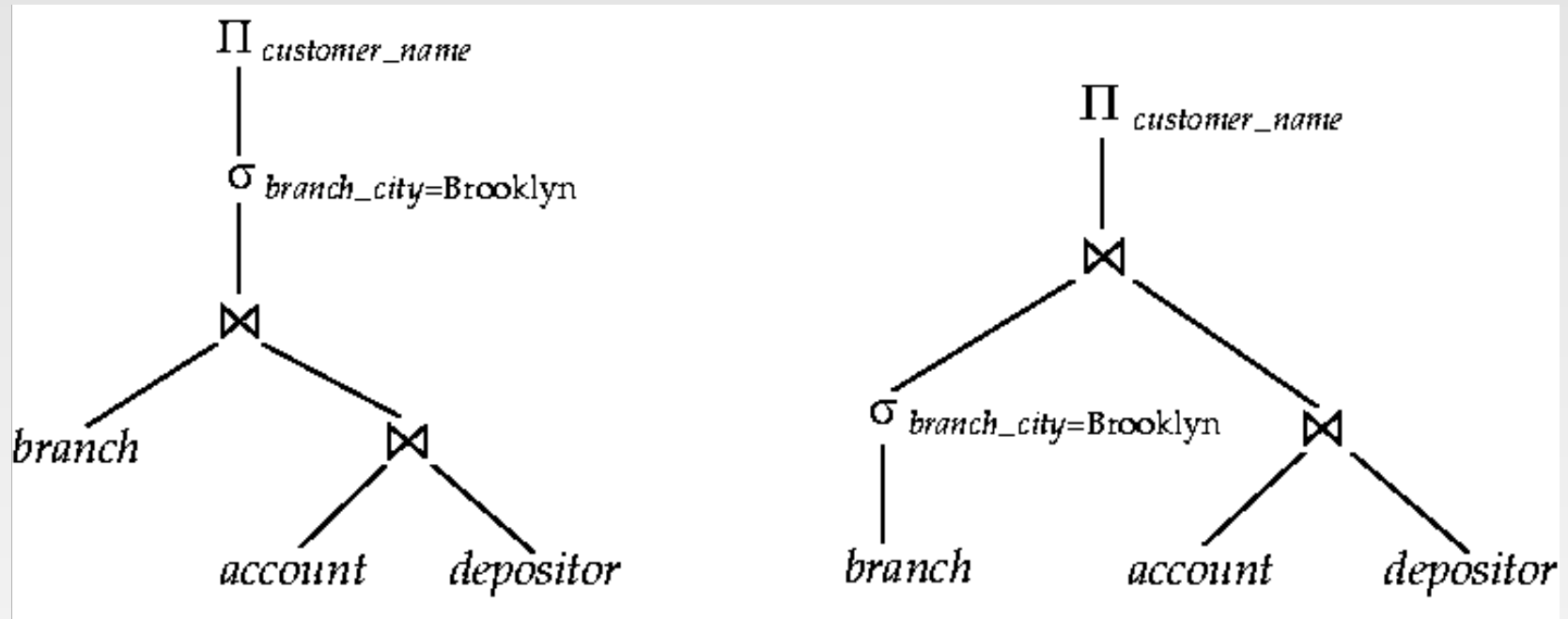
- Introduction
- Transformation of Relational Expressions





Introduction

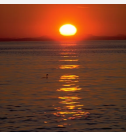
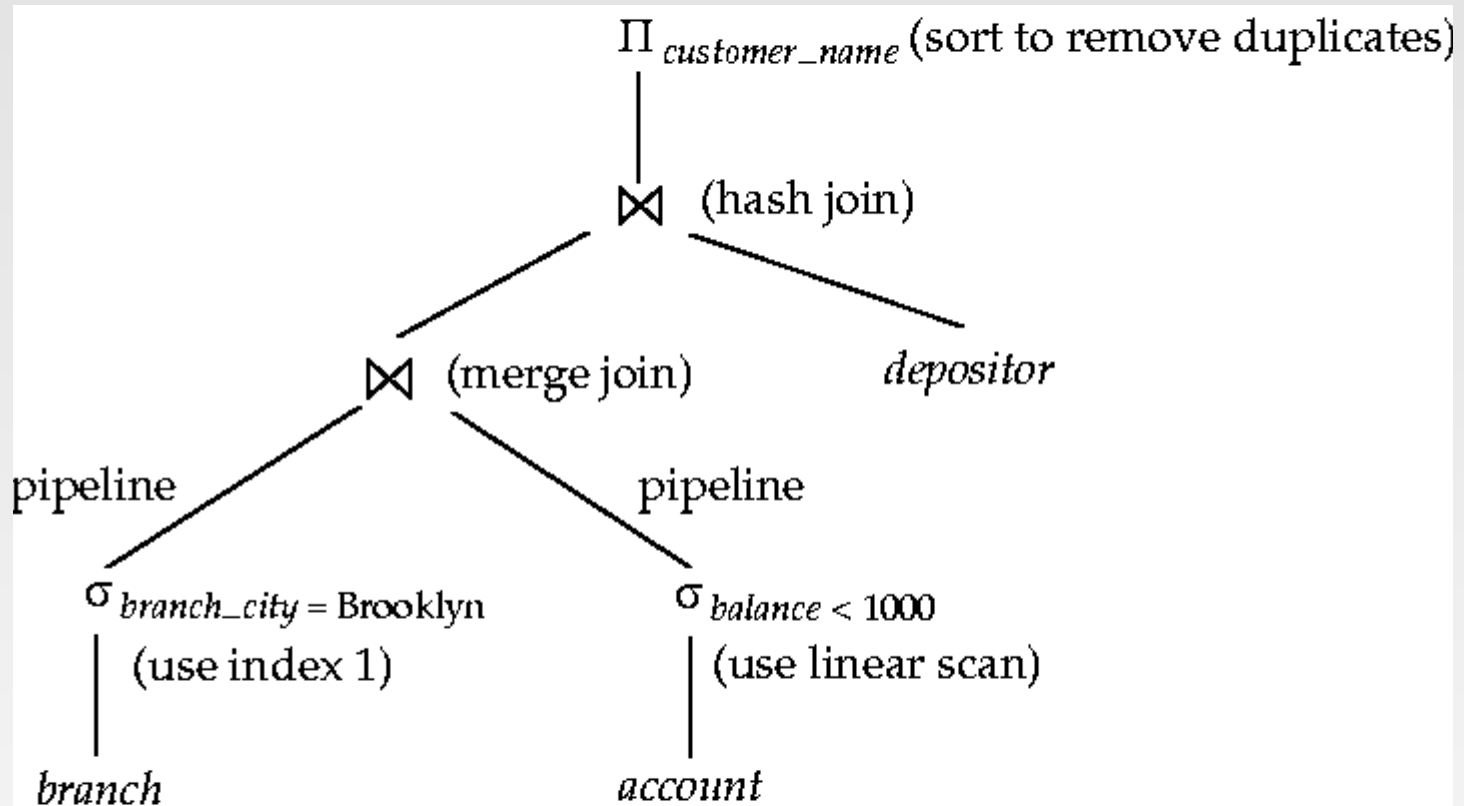
- Alternative ways of evaluating a given query
 - Equivalent expressions
 - Different algorithms for each operation





Introduction (Cont.)

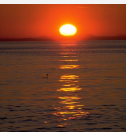
- An **evaluation plan** defines exactly what algorithm is used for each operation, and how the execution of the operations is coordinated.





Introduction (Cont.)

- Cost difference between evaluation plans for a query can be enormous
 - E.g. seconds vs. days in some cases
- Steps in **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**
- Estimation of plan cost based on:
 - Statistical information about relations. Examples:
 - ▶ number of tuples, number of distinct values for an attribute
 - Statistics estimation for intermediate results
 - ▶ to compute cost of complex expressions
 - Cost formulae for algorithms, computed using statistics





Generating Equivalent Expressions

Database System Concepts 5th Ed.

©Silberschatz, Korth and Sudarshan

See www.db-book.com for conditions on re-use





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
 - 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 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.

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

2. Selection operations are commutative.

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

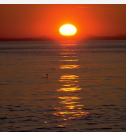
3. Only the last in a sequence of projection operations is needed, the others can be omitted.

$$\Pi_{L_1}(\Pi_{L_2}(\dots(\Pi_{L_n}(E))\dots)) = \Pi_{L_1}(E)$$

1. Selections can be combined with Cartesian products and theta joins.

$\alpha.$ $\sigma_{\theta}(E_1 \times E_2) = E_1 \bowtie_{\theta} E_2$

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





Equivalence Rules (Cont.)

5. Theta-join operations (and natural joins) are commutative.

$$E_1 \bowtie_{\theta} E_2 = E_2 \bowtie_{\theta} E_1$$

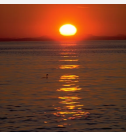
6. (a) Natural join operations are associative:

$$(E_1 \bowtie E_2) \bowtie E_3 = E_1 \bowtie (E_2 \bowtie E_3)$$

- (b) Theta joins are associative in the following manner:

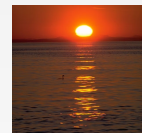
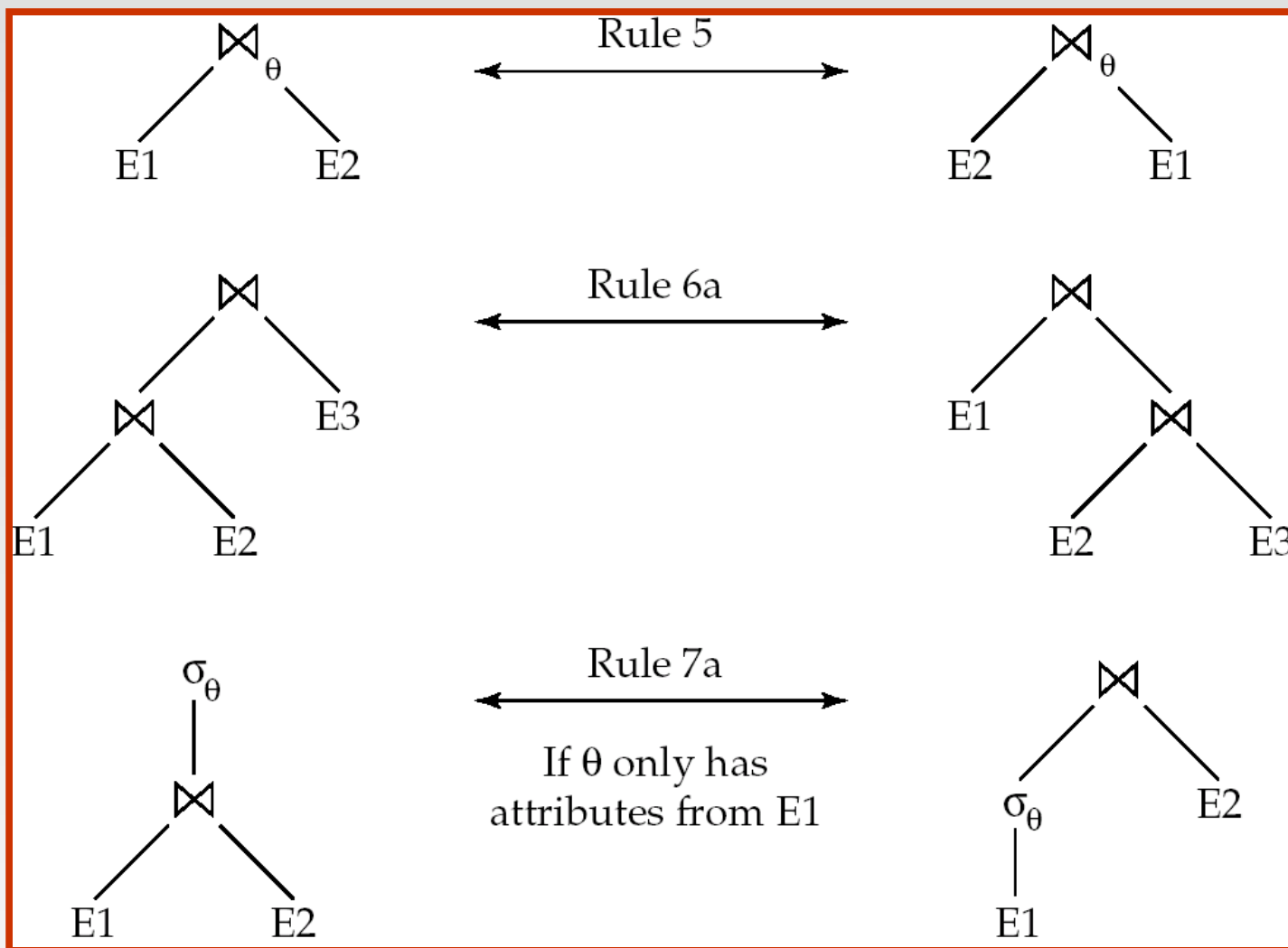
$$\bowtie (E_1 \bowtie_{\theta_1} E_2) \bowtie_{\theta_2 \wedge \theta_3} E_3 = E_1 \bowtie_{\theta_1 \wedge \theta_3} (E_2 \bowtie_{\theta_2} E_3)$$

where θ_2 involves attributes from only E_2 and E_3 .





Pictorial Depiction of Equivalence Rules





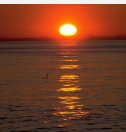
Equivalence Rules (Cont.)

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 (Cont.)

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

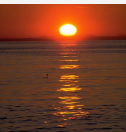
(a) if θ involves only attributes from $L_1 \cup L_2$:

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

(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 θ , but are not in $L_1 \cup L_2$, and
- let L_4 be attributes of E_2 that are involved in join condition θ , 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 (Cont.)

1. 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).

2. Set union and intersection are associative.

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

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

1. The selection operation distributes over \cup , \cap and $-$.

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

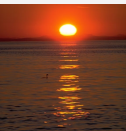
and similarly for \cup and \cap in place of $-$

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 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 7a.

$\Pi_{customer_name}((\sigma_{branch_city = "Brooklyn"}(branch)) \bowtie (account \bowtie depositor))$

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





Example with Multiple Transformations

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

$$\Pi_{customer_name}(\sigma_{branch_city = "Brooklyn" \wedge balance > 1000}((branch \bowtie_{account} depositor)))$$

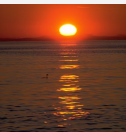
- Transformation using join associatively (Rule 6a):

$$\Pi_{customer_name}((\sigma_{branch_city = "Brooklyn" \wedge balance > 1000}(branch \bowtie_{account} depositor)))$$

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

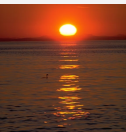
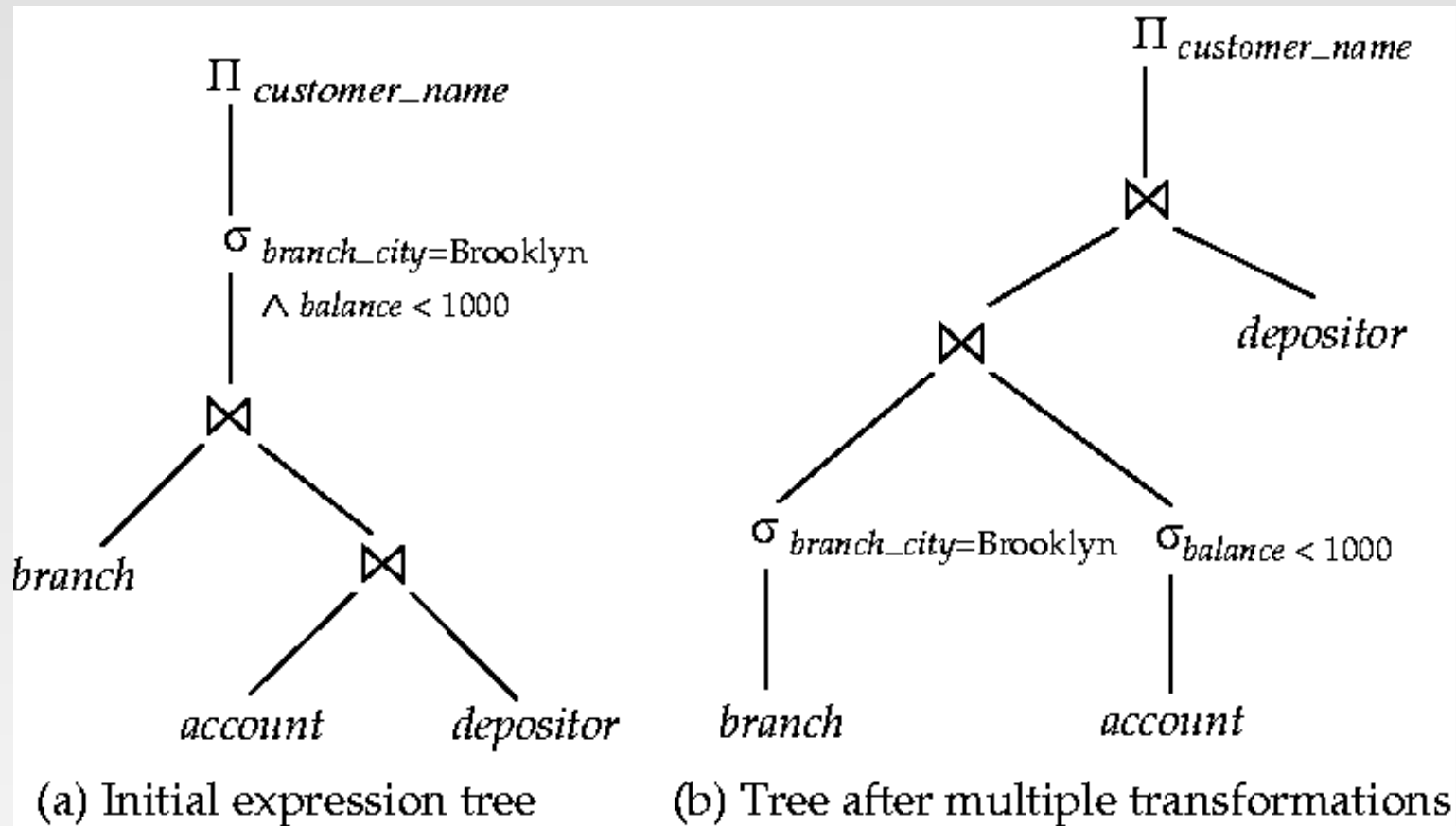
$$\sigma_{branch_city = "Brooklyn"}(branch) \bowtie_{balance > 1000} \sigma_{balance > 1000}(account)$$

- Thus a sequence of transformations can be useful





Multiple Transformations (Cont.)





Transformation Example: Pushing Projections

$\Pi_{customer_name}((\sigma_{branch_city = \text{"Brooklyn"}}(branch) \bowtie_{\cancel{a}} \cancel{account}) \bowtie_{\cancel{d}} \cancel{depositor})$

- When we compute

$(\sigma_{branch_city = \text{"Brooklyn"}}(branch) \bowtie_{\cancel{a}} account)$

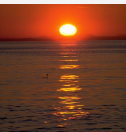
we obtain a relation whose schema is:

$(branch_name, branch_city, assets, account_number, balance)$

- Push projections using equivalence rules 8a and 8b; eliminate unneeded attributes from intermediate results to get:

$\Pi_{customer_name}((\Pi_{account_number}(\sigma_{branch_city = \text{"Brooklyn"}}(branch) \bowtie_{\cancel{a}} account) \bowtie_{\cancel{d}} \cancel{depositor}))$

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





End of Chapter

Database System Concepts 5th Ed.

©Silberschatz, Korth and Sudarshan

See www.db-book.com for conditions on re-use

