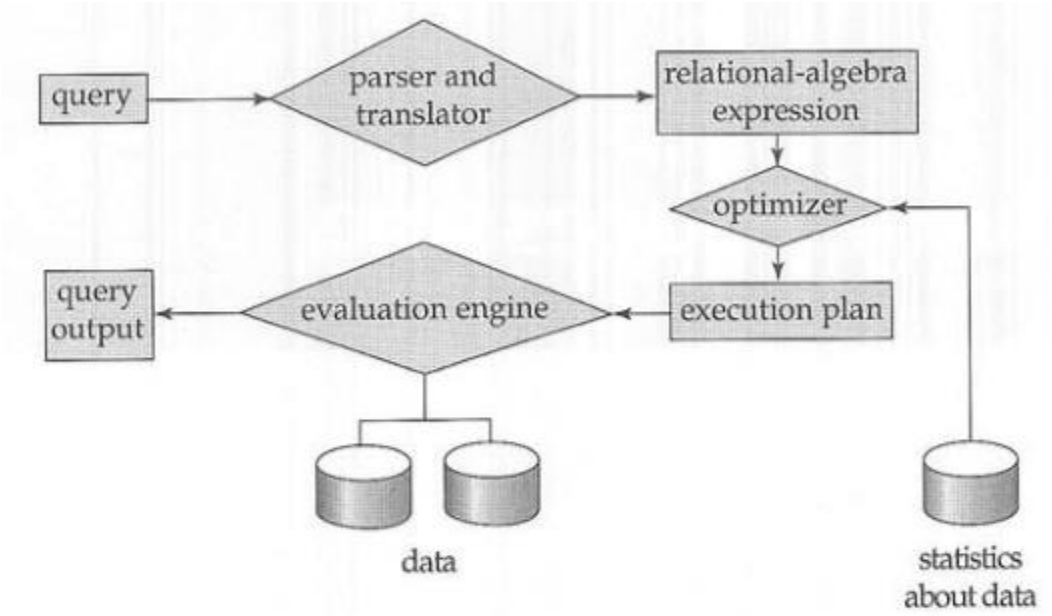
Query Equivalent

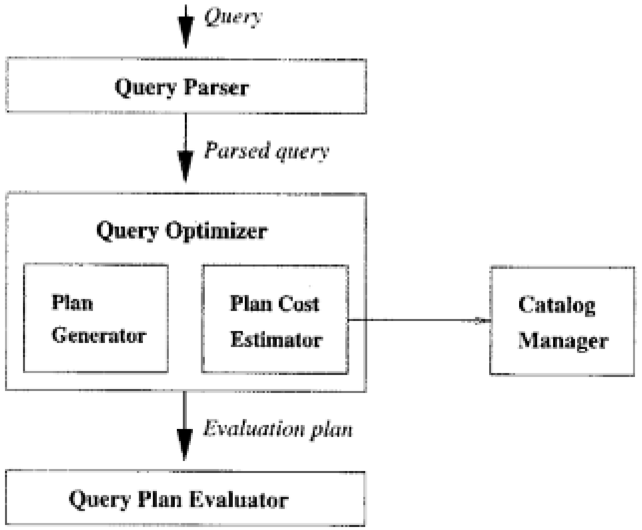
(Query processing Engine)

Adapted from

Chapter 14 – Database System Concept, Silberschatz, Korth, Sudarshan, 5th ed.
Chapter 15 – Database Management System, Ramakrishnan, Gehrke, 3rd ed



Query Evaluation Engine



Sailors(*sid:* integer, *sname:* string, *rating:* integer, *age:* real)
Reserves(*sid:* integer, *bid:* integer, *day:* dates, *rname:* string)

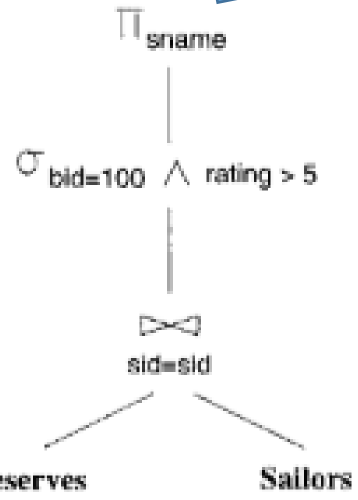
Database

Query

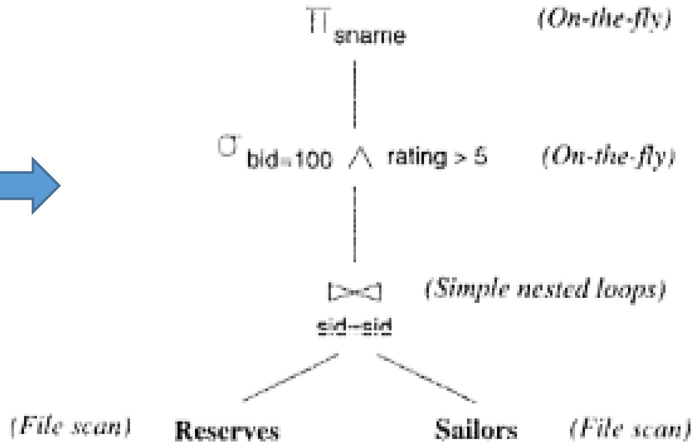
```
SELECT S.sname
FROM   Reserves R, Sailors S
WHERE  R.sid = S.sid
       AND R.bid = 100 AND S.rating > 5
```

$$\pi_{sname}(\sigma_{bid=100 \wedge rating>5}(Reserves \bowtie_{sid=sid} Sailors))$$

Relational Algebra

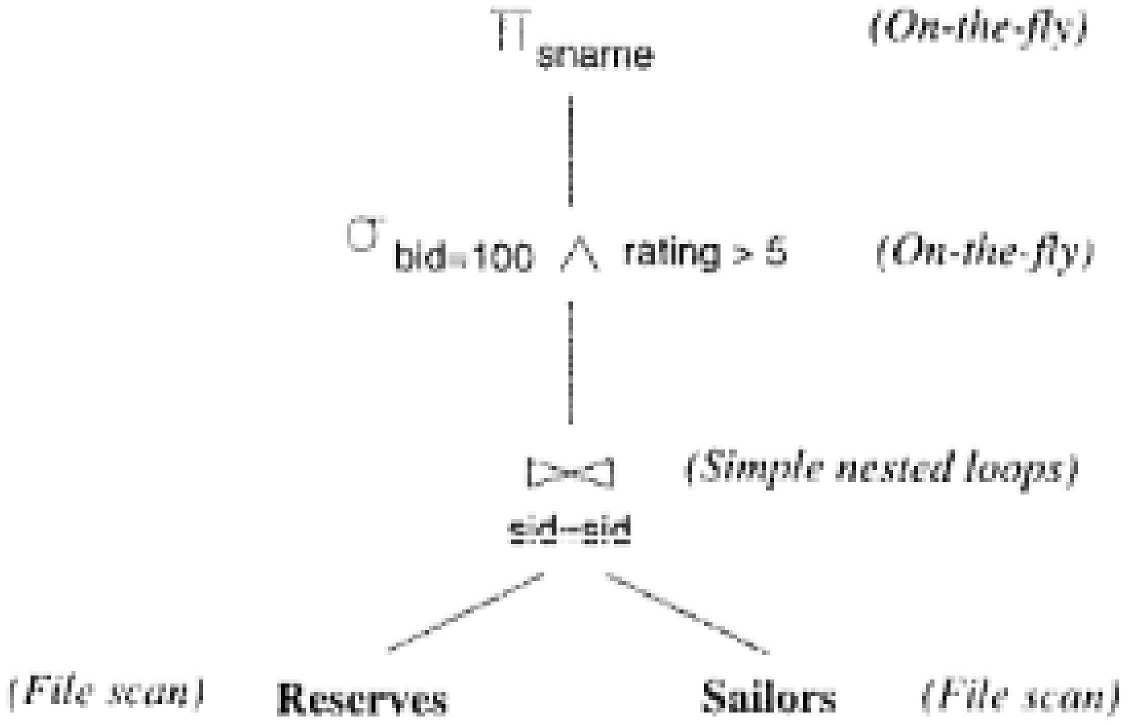


Relational Algebra tree



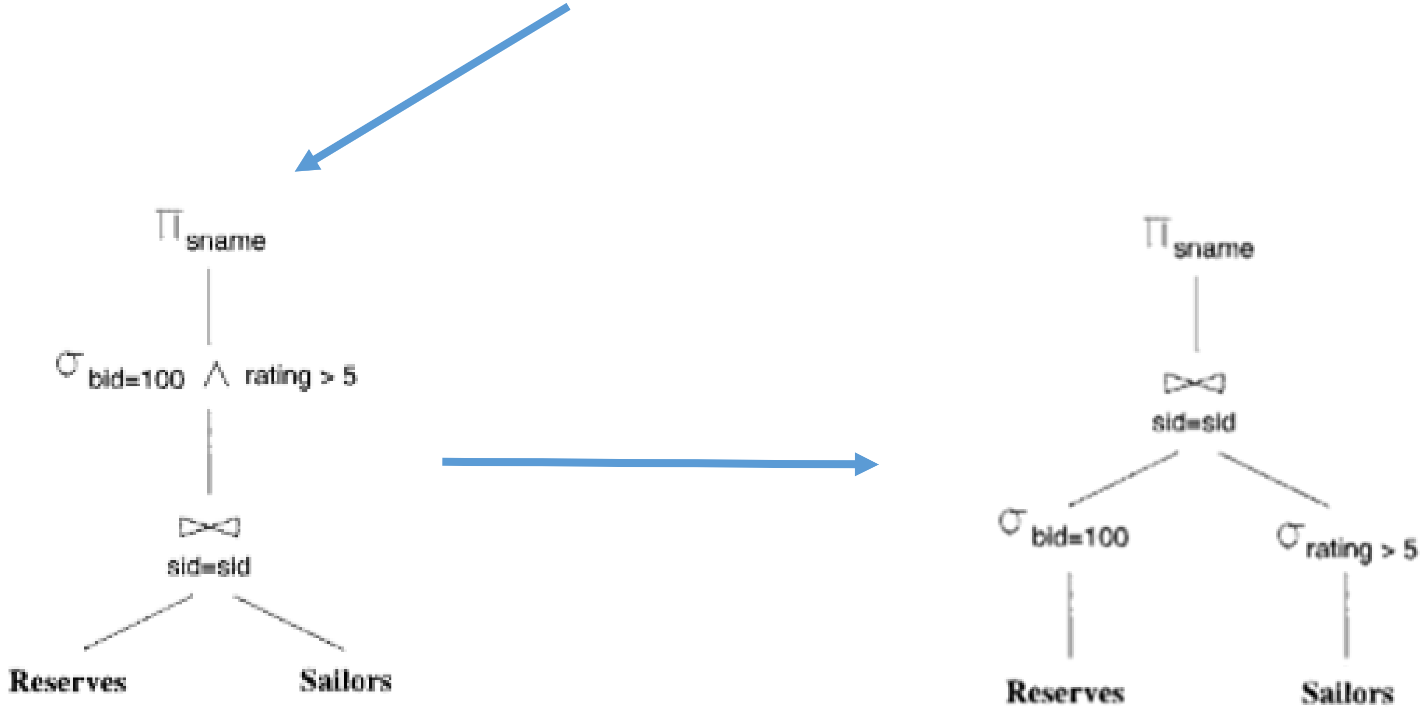
Query Evaluation Plan

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



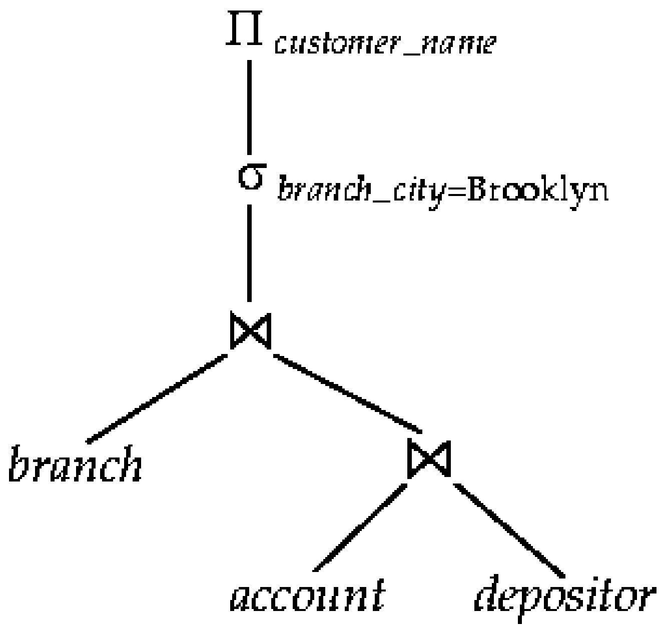
A query can be served with multiple transformations (execution rules) providing the same results

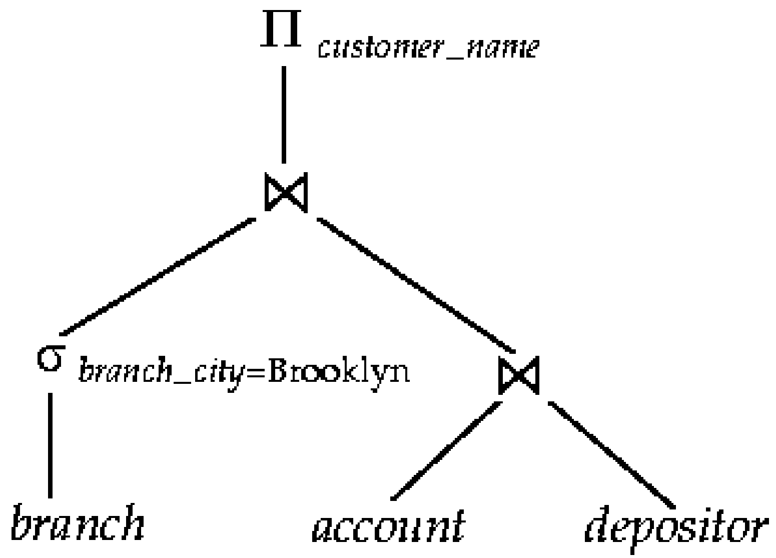
$$\pi_{sname}(\sigma_{bid=100 \wedge rating>5}(Reserves \bowtie_{sid=sid} Sailors))$$



A query can be served with various execution plans providing the same results

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





- 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**

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
  - 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 (**cascading of selection**).

$$\sigma_{c_1 \wedge c_2 \wedge \ldots c_n}(R) \equiv \sigma_{c_1}(\sigma_{c_2}(\ldots(\sigma_{c_n}(R))\ldots))$$

Equivalence Rules

1. Conjunctive selection operations can be deconstructed into a sequence of individual selections (**cascading of selection**).

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

EiD	Name	Salary	Age
01	Ram	30000	35
02	Shyam	10000	25
03	Kumar	25000	30
04	Ram	50000	45
05	John	75000	50
05	Jack	5000	20

→ $\sigma_{\text{Salary>20000 AND Age < 40}}(R)$ →

EiD	Name	Salary	Age
01	Ram	30000	35
03	Kumar	25000	30

Equivalence Rules

1. Conjunctive selection operations can be deconstructed into a sequence of individual selections (**cascading of selection**).

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

EiD	Name	Salary	Age
01	Ram	30000	35
02	Shyam	10000	25
03	Kumar	25000	30
04	Ram	50000	45
05	John	75000	50
05	Jack	5000	20

$\sigma_{\text{Salary>20000 AND Age < 40}}(\text{R})$

EiD	Name	Salary	Age
01	**Ram**	**30000**	**35**
03	**Kumar**	**25000**	**30**

$\sigma_{\text{Salary>20000}}(\sigma_{\text{Age < 40}}(\text{R}))$

$\sigma_{\text{Salary>20000}}($

EiD	Name	Salary	Age
01	Ram	30000	35
02	Shyam	10000	25
03	Kumar	25000	30
05	Jack	5000	20

Equivalence Rules

2. Selection operations are **commutative**.

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

Equivalence Rules

2. Selection operations are **commutative**.

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

EiD	Name	Salary	Age
01	Ram	30000	35
02	Shyam	10000	25
03	Kumar	25000	30
04	Ram	50000	45
05	John	75000	50
05	Jack	5000	20

$\sigma_{\text{Salary>20000 AND Age < 40}}(R)$

EiD	Name	Salary	Age
01	**Ram**	**30000**	**35**
03	**Kumar**	**25000**	**30**

$\sigma_{\text{Salary>20000}}(\sigma_{\text{Age < 40}}(R))$

$\sigma_{\text{Salary>20000}}($

EiD	Name	Salary	Age
01	Ram	30000	35
02	Shyam	10000	25
03	Kumar	25000	30
05	Jack	5000	20

$\sigma_{\text{Age < 40}}(\sigma_{\text{Salary>20000}}(R))$

$\sigma_{\text{Age < 40}}($

EiD	Name	Salary	Age
01	Ram	30000	35
03	Kumar	25000	30
04	Ram	50000	45
05	John	75000	50

Equivalence Rules

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

$$\Pi_{L_1}(\Pi_{L_2}(\ldots(\Pi_{Ln}(E))\ldots)) = \Pi_{L_1}(E)$$

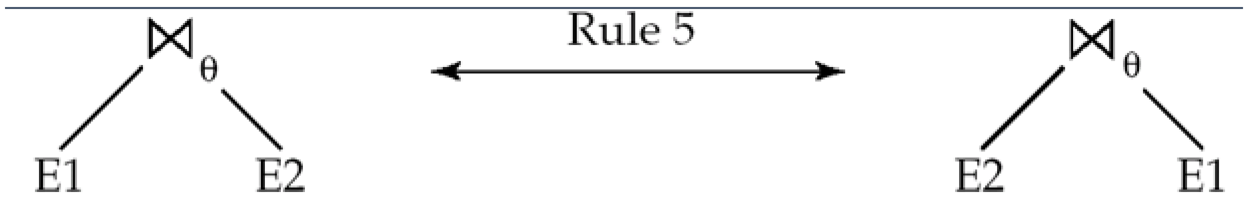
Equivalence Rules

4. Selections can be combined with Cartesian products and theta joins.
   a. $\sigma_{\theta}(E_1 \times E_2) = E_1 \bowtie_{\theta} E_2$
   b. $\sigma_{\theta 1}(E_1 \bowtie_{\theta 2} E_2) = E_1 \bowtie_{\theta 1 \wedge \theta 2} E_2$

Equivalence Rules

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

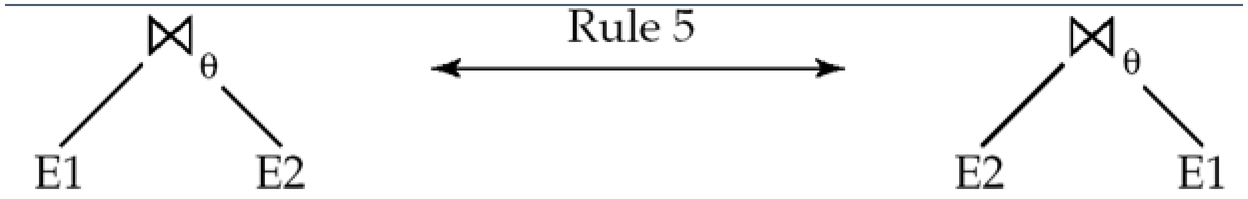
$$E_1 \bowtie_\theta E_2 = E_2 \bowtie_\theta E_1$$



Equivalence Rules

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

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



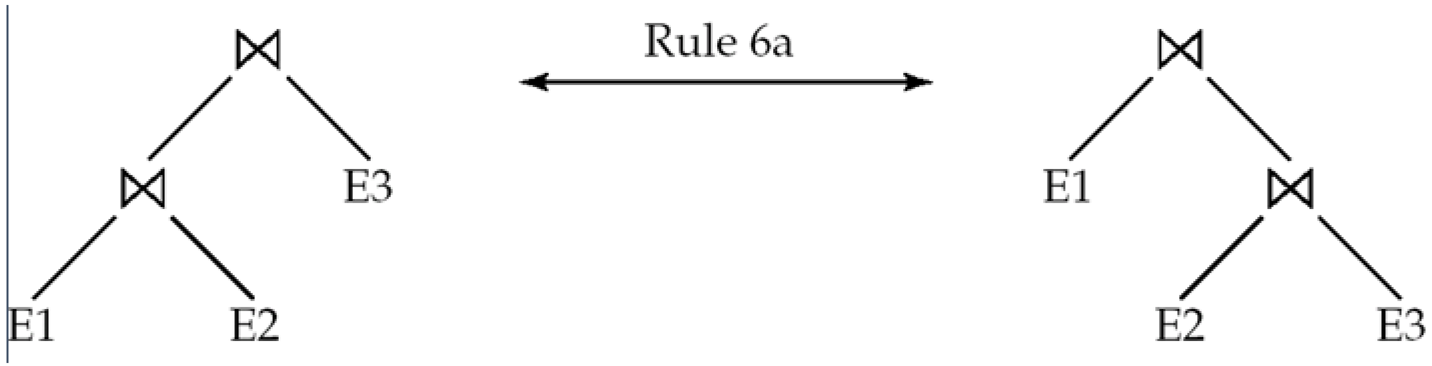
Cartesian Product also holds the following

$$R \times S \equiv S \times R$$

Equivalence Rules (Cont.)

6. (a) Natural join operations are **associative**:

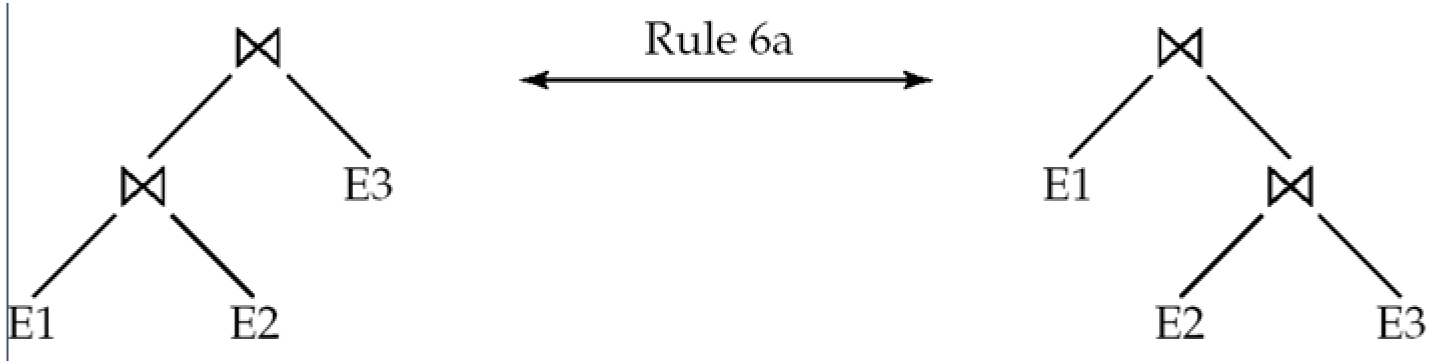
$$(E_1 \bowtie E_2) \bowtie E_3 = E_1 \bowtie (E_2 \bowtie E_3)$$



Equivalence Rules (Cont.)

6. (a) Natural join operations are **associative**:

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



Natural Join also holds the following

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

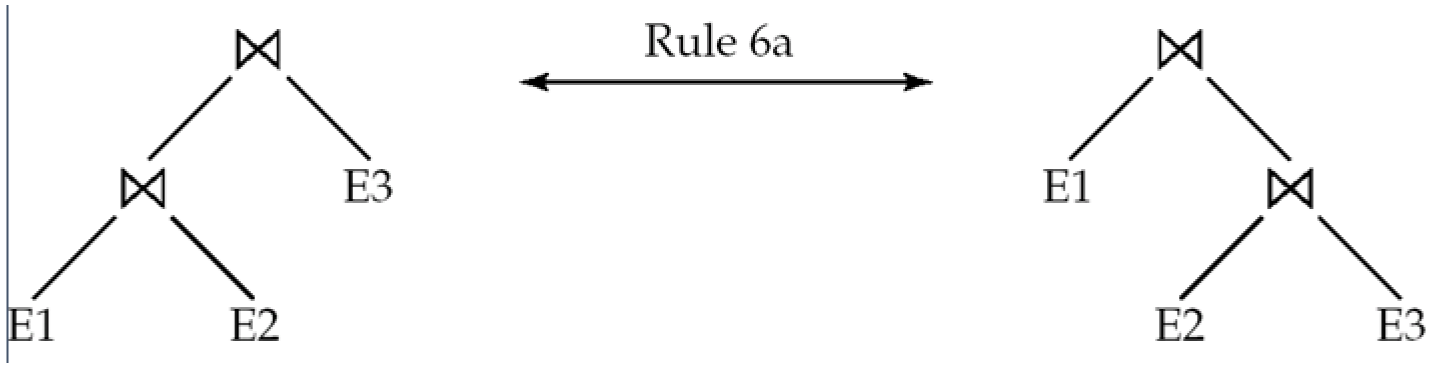
Equivalence Rules (Cont.)

6. (a) Natural join operations are **associative**:

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

Cartesian Product also holds the following

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



Natural Join also holds the following

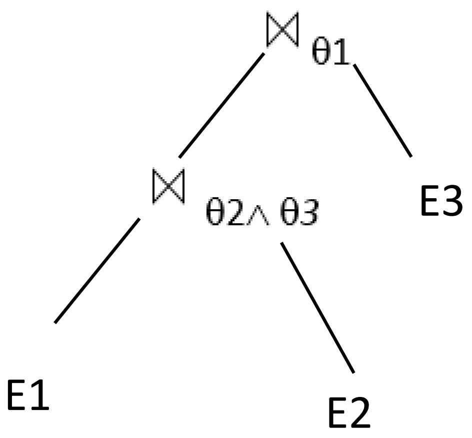
$$R \bowtie (S \bowtie T) \equiv (T \bowtie R) \bowtie S$$

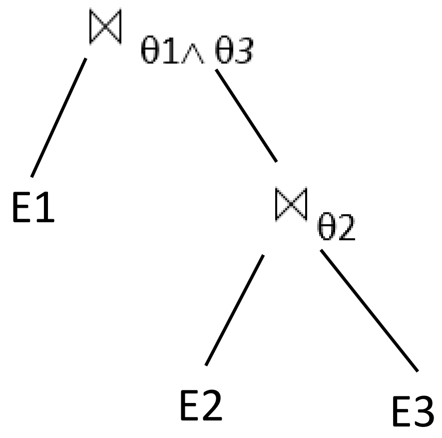
Equivalence Rules (Cont.)

6.(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.

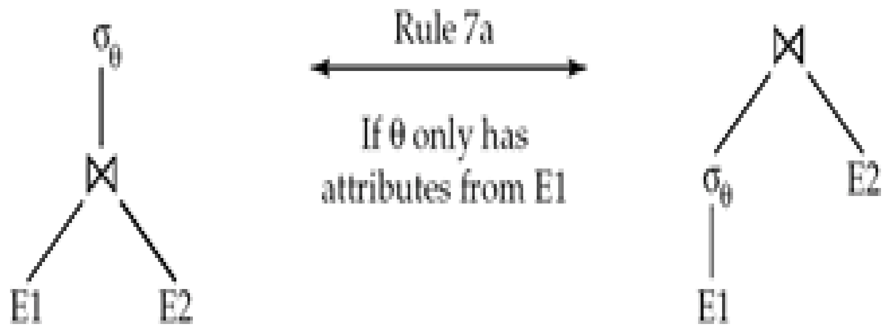


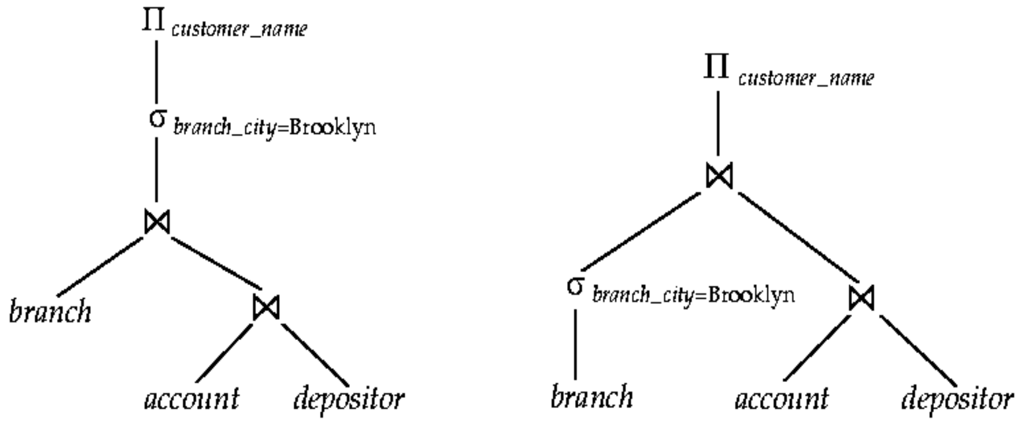


Equivalence Rules (Cont.)

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.

$$\sigma_{\theta 0}(E_1 \bowtie_\theta E_2) = (\sigma_{\theta 0}(E_1)) \bowtie_\theta E_2$$



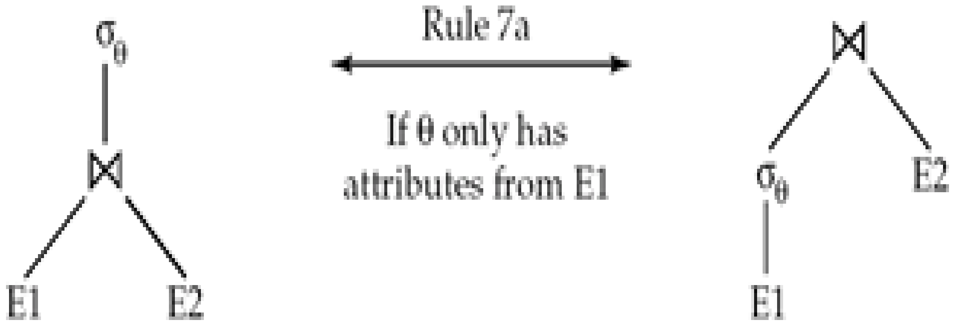


Equivalence Rules (Cont.)

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.

$$\sigma_{\theta 0}(E_1 \bowtie_{\theta} E_2) = (\sigma_{\theta 0}(E_1)) \bowtie_{\theta} E_2$$

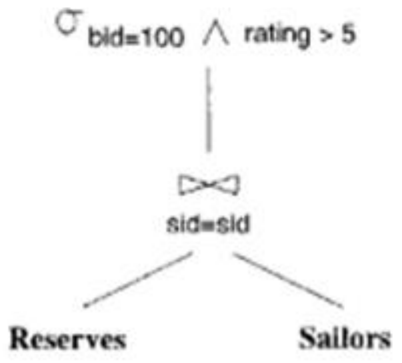
Also true for Cartesian Product

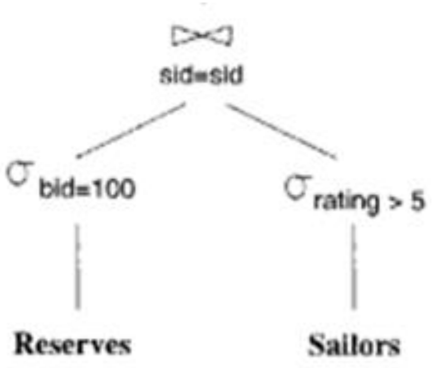


Equivalence Rules (Cont.)

7. The selection operation distributes over the theta join operation under the following two conditions:
   (b) When θ_1 involves only the attributes of E_1 and θ_2 involves only the attributes of E_2.

$$\sigma_{\theta 1 \wedge \theta 2}(E_1 \bowtie_\theta E_2) = (\sigma_{\theta 1}(E_1)) \bowtie_\theta (\sigma_{\theta 2}(E_2))$$





Projection over Selection

Can we perform?

$$\pi_a(\sigma_c(R)) \equiv \sigma_c(\pi_a(R))$$

Projection over Selection

Can we perform?

$$\pi_a(\sigma_c(R)) \equiv \sigma_c(\pi_a(R))$$

Yes, if ***c*** is included in ***a***

Projection over Selection

Can we perform?

$$\Pi_{\theta1,\theta2}(E_1 \bowtie_\theta E_2) = (\Pi_{\theta1}(E_1)) \bowtie_\theta (\Pi_{\theta2}(E_2))$$

Projection over Selection

Can we perform?

$$\Pi_{\theta 1,\theta 2}(E_1 \bowtie_\theta E_2) = (\Pi_{\theta 1}(E_1)) \bowtie_\theta (\Pi_{\theta 2}(E_2))$$

$$\Pi_{\theta 1,\theta 2}(E_1 \bowtie_\theta E_2) = (\Pi_{\theta 1,\theta 2}(E_1)) \bowtie_\theta (\Pi_{\theta 1,\theta 2}(E_2))$$

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 $L_1 \cup L_2$.

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

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**.

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

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 $\cap$ 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

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

Example:

branch(branch_name, branch_city, assets),
account(account_number, branch_name, balance)
depositor(customer_name,account_number)

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

$$\Pi_{customer\_name}(\sigma_{branch\_city = \text{“Brooklyn”}}(branch \bowtie (account \bowtie depositor)))$$

- Transformation using rule 7a.

$$\Pi_{customer\_name}((\sigma_{branch\_city = \text{“Brooklyn”}}(branch)) \bowtie (account \bowtie depositor))$$

- Can also be transformed as

$$\sigma_{branch\_city = \text{“Brooklyn”}}(\Pi_{customer\_name,\ branch\_city}(branch \bowtie (account \bowtie depositor)))$$

Example with Multiple Transformations

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

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

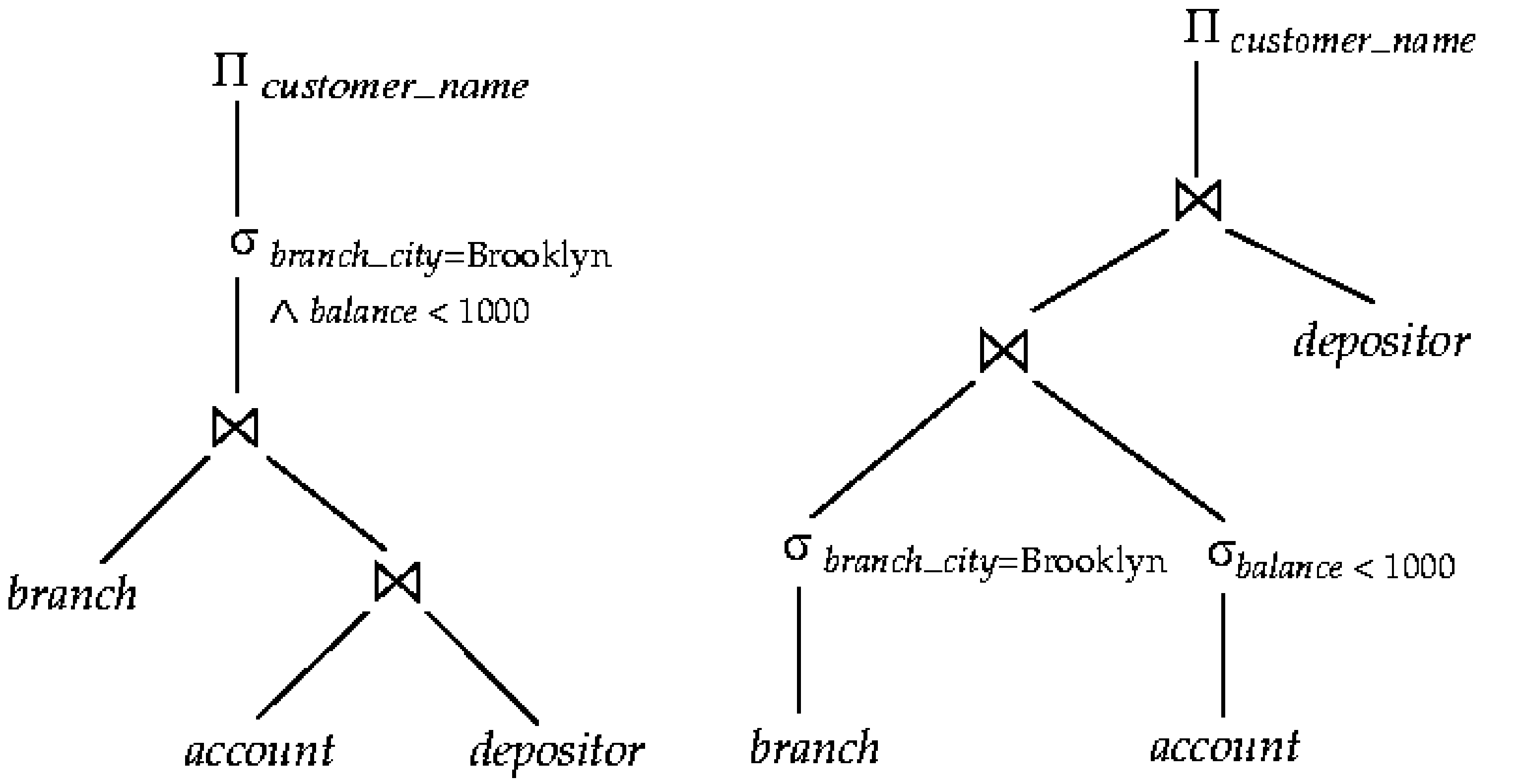
- Transformation using join associatively (Rule 6a):

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

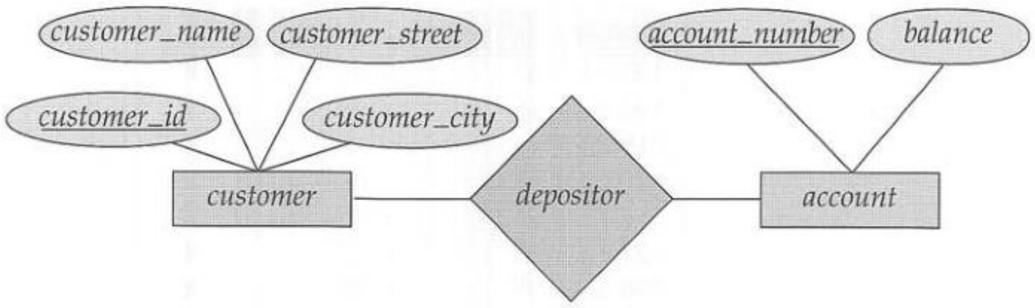
- Second form provides an opportunity to apply the “perform selections early” rule, resulting in the subexpression

$$\sigma_{branch\_city = \text{“Brooklyn”}}(branch) \bowtie \sigma_{balance > 1000}(account)$$

Multiple Transformations (Cont.)



(a) Initial expression tree (b) Tree after multiple transformations



$$\Pi_{customer\_name}(\sigma_{branch\_city = \text{“Brooklyn”}}(branch \bowtie (account \bowtie depositor)))$$

$$\Pi_{customer\_name}((\sigma_{branch\_city = \text{“Brooklyn”}}(branch) \bowtie account) \bowtie depositor)$$

$$\Pi_{customer\_name}((\Pi_{account\_number}((\sigma_{branch\_city = \text{“Brooklyn”}}(branch) \bowtie account)) \bowtie depositor)$$