数据库系统原理

Database System Principle

邵蓥侠

Email：shaoyx@bupt.edu.cn

北京邮电大学计算机学院

计算机应用技术中心

PART 6

QUERY PROCESSING AND OPTIMIZATION

Chapter 16

Query Optimization

Main parts in Chapter 16

- § 16.1 Overview
  - why optimization needed
- § 16.2 Transformation of relational expressions
  - equivalence rules
- § 16.4 Choice of evaluation plans — Query optimization
  - cost-based optimization, § 16.4.1/16.4.2
  - ***heuristic optimization,*** § 16.4.3

§ 16.1 Overview

- A SQL query statement may corresponds to several equivalent expressions (*or* query evaluation plans), each of which may have different costs
- ***Query optimization*** is needed to choice an ***efficient*** query-evaluation plan with ***lower*** costs

- E.g.1. select *
  from r, s
  where r.A = s.A

$\boldsymbol{\sigma}_{r.A=s.A}\,(r \times s)$ is much slower than $r \bowtie s$

Overview (cont.)

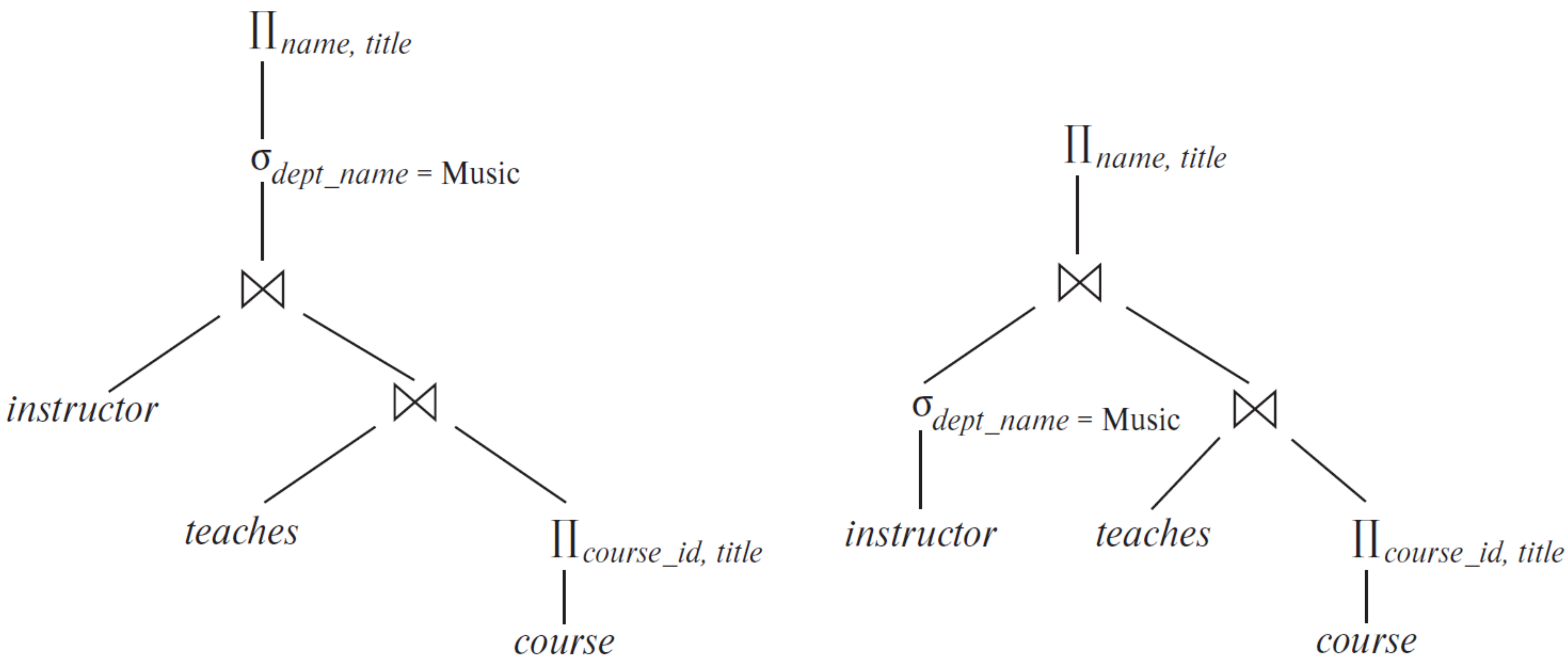
- E.g.2 Find the *names* of all *instructors* in the *Music* department together with the course title of all the courses that the instructors teach

$$\Pi_{name,title}\ (\sigma_{dept\_name = \text{“Music”}}\ (instructor\ \bowtie\ (teaches\ \bowtie\ \Pi_{course\_id,title}(course))))$$

$$\Pi_{name,title}\ ((\sigma_{dept\_name = \text{“Music”}}\ (instructor))\bowtie\ (teaches\ \bowtie\ \Pi_{course\_id,title}(course)))$$

Overview (cont.)

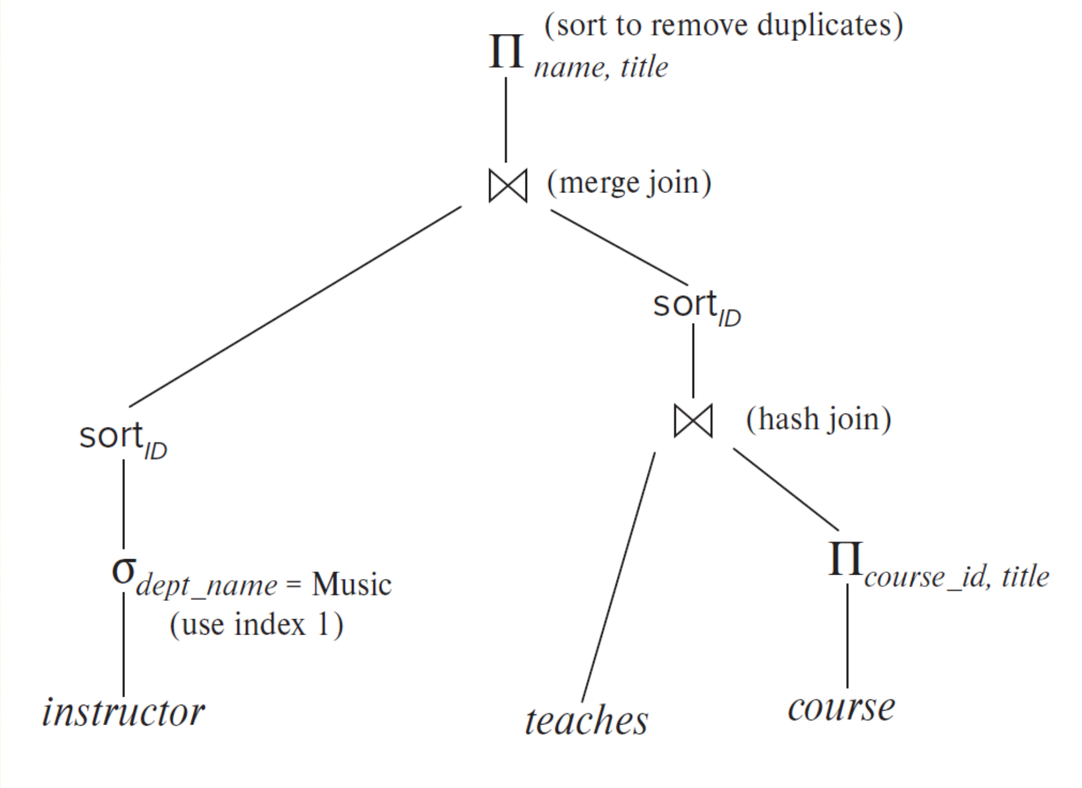
- Alternative ways of evaluating a given query
  - Equivalent expressions
  - Different algorithms for each operation



(a) Initial expression tree (b) Transformed expression tree

Overview (cont.)

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



Overview (cont.)

- Cost difference between evaluation plans for a query can be enormous
  - E.g. seconds vs. days in some cases

Overview (cont.)

- The procedures of optimization
  - generating the equivalent expressions/query trees, by transforming of relational algebra expressions according to equivalence rules in § 16.2
  - generating alternative *evaluation plans,* by annotating the resultant equivalent expressions with implementation algorithms for each operations in the expressions
  - choosing the optimal (that is, cheapest) or near-optimal plan based on the **estimated cost**, by
    - cost-based optimization
    - ***heuristic optimization***

Overview (cont.)

- The cost of operations is needed to estimate 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

§ 16.2 Transformation of Relational Expressions

- **Definition.** Two relational algebra expressions are equivalent, if on every legal database instances, the two expression generate the same set of tuples

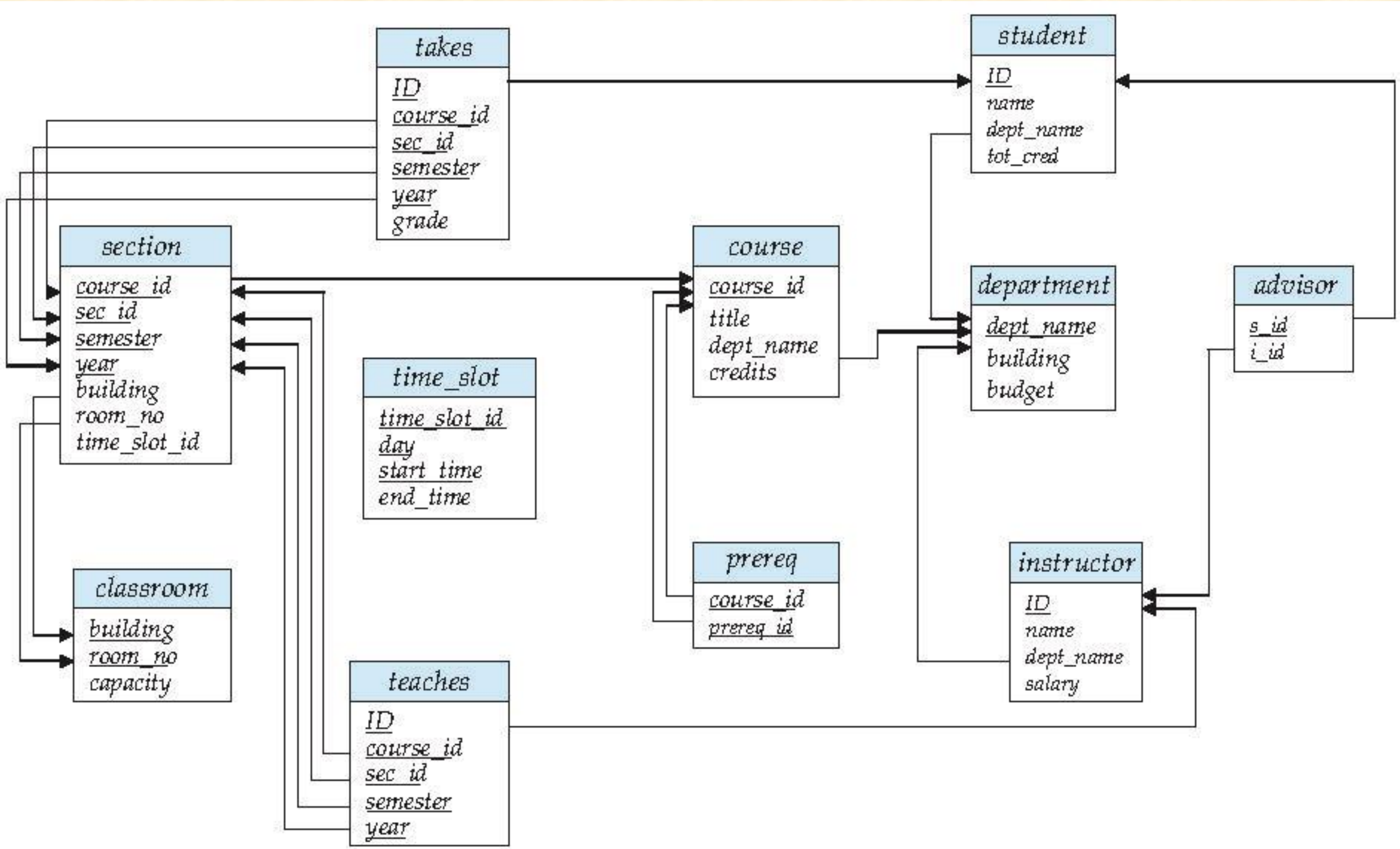


Fig.16.0.1 Schema diagram for the University database

Equivalence Rules (cont.)

- Rule1.(选择串接律, 将1个选择操作分解为2个选择操作) Conjunctive（合取） selection operations can be deconstructed into a sequence of individual selections
- 合取选择运算可分解为单个选择运算的序列。

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

  - e.g. refer to

- Rule2. (选择交换律) Selection operations are *commutative*（交换律）：

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

Equivalence Rules (cont.)

- Rule3. (投影串接律) Only the final operation in a sequence of project operations is needed

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

  - NOTE: it is required that $L_1 \subseteq L_2 \subseteq \ldots . \subseteq L_n$
  - e.g. $\Pi_{\{ID\}}\ \{\ \Pi_{\{ID,\ name\}}\ (instructor)\ \}$
    $= \Pi_{\{ID\}}\ (instructor)$

Equivalence Rules (cont.)

- **Rule4**. **(以连接**操作代替**选择**和**笛卡尔乘积)** Selections can be combined with Cartesian products and theta joins

  a. $\sigma_\theta(E_1 \times E_2) = E_1 \bowtie_\theta E_2$

  note: definition of θ join

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

  - e.g. $\sigma_{\textit{dept-name}=\text{“}Music\text{”}}(\textit{instructor} \bowtie \textit{teaches}) =$

  $\textit{instructor} \bowtie_{(\textit{dept-name}=\text{“}Music\text{”}) \wedge (\text{instructor.ID=teaches.ID})} \textit{teaches}$

- By Rule4, the number of operations can be reduced, and the costs of the right-hand expressions are less than that of the left-hand expressions
  - often used in heuristic query optimizing

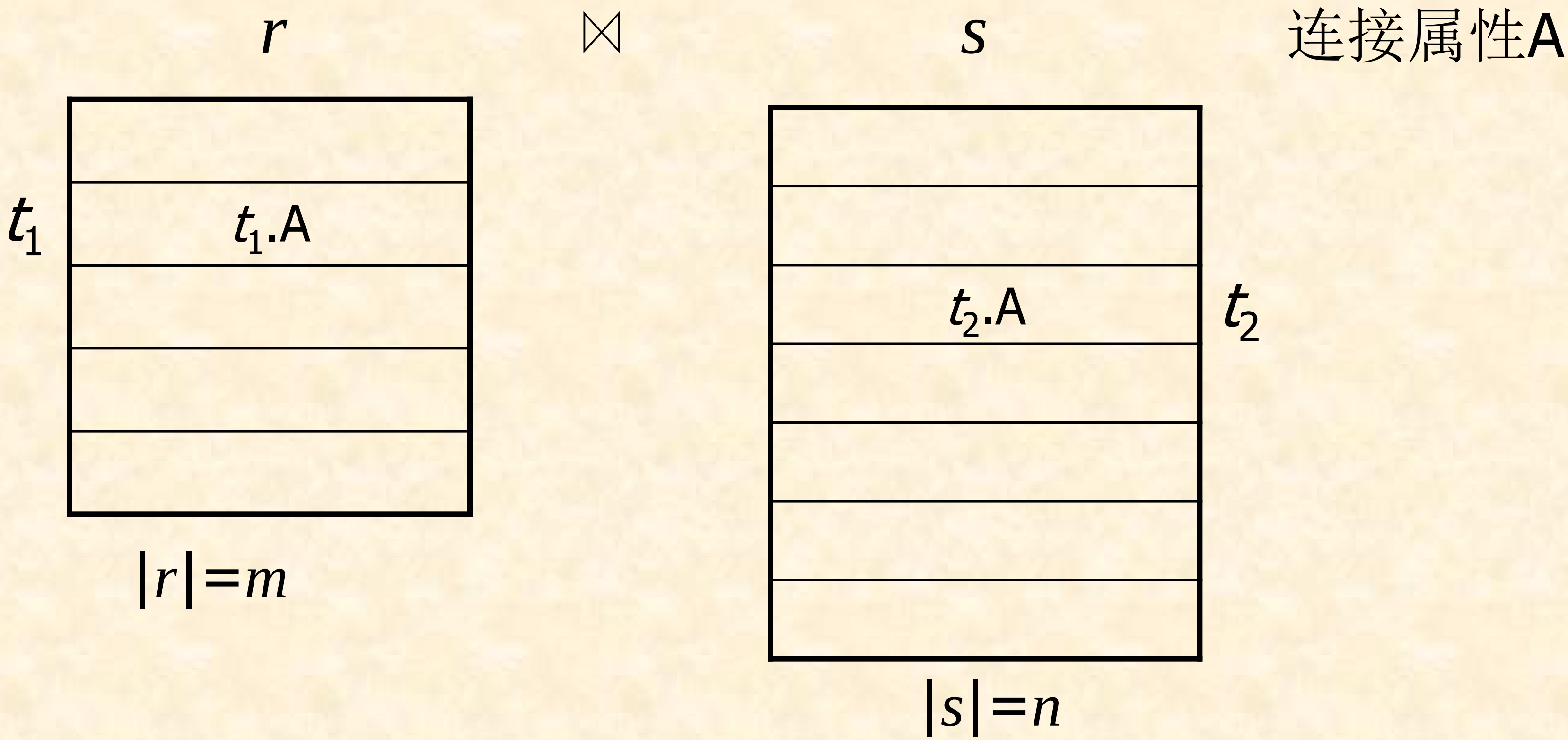
Equivalence Rules (cont.)

- !!! Rule5 (连接操作可交换) Theta join operations are *commutative*

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

  - Fig. 16.3

  - ***!! the expression with smaller size should be arranged as the left one in the operation***



原理：　针对r中元组t_1，检查s中的元组t_2.A，t_1.A=t_2.A ?

方法：

For r中每个元组t_1，　　　　　　　　//*扫描

　按照t_1.A，查找s中满足t_1.A=t_2.A的元组t_2，

　合并元组t_1和t_2

- 假设

  1. $|r|=m$，$|s|=n$

  2. $m<n$, $n/m = k > 1$

- 给定r中的元组t_1，采用B/B$^+$树索引、二分搜索机制，根据t_1.A，查找s中满足$t_1.A=t_2.A$的元组t_2，所需cost正比于树高，为：

$$O(\lg n)$$

$r \bowtie s$的cost为：　$O(m + m\lg n)$

$s \bowtie r$的cost为：　$O(n + n\lg m)$

- 如果没有索引：

$O(m + m * n)$　vs　$O(n + n * m)$

$m + m\lg n$ vs $n + n\lg m$

- $m<n$, $n/m = k > 1$, $n = k*m$
- $n\lg m - m\lg n$

$$= k*m\lg m - m(\lg k*m)$$

$$= k*m\lg m - m\lg k - m\lg m$$

$$= (k-1)m\lg m - m\lg k$$

- 一般情况下，$(k-1)m\lg m > m\lg k$
- E.g. $|r|=m=500$， $|s|=n=1000$, $k = n/m = 2$

$(k-1)m\lg m = 500\lg 500 > m\lg k = 500\lg 2$

r: *depositor*, $|r| = 7$

s: *account*, $|s| = 10$

表 - dbo.account 表 - dbo.depositor

customername	accountnumber
Hayes	A-102
Johnson	A-101
Johnson	A-201
Jones	A-217
Lindsay	A-222
Simith	A-215
Turner	A-305
NULL	*NULL*

表 - dbo.account 表 - dbo.depositor

accountnumber	branchname	balance
A-101	Downtown	500
A-102	Perryridge	400
A-105	Perryridge	500
A-201	Brigh	900
A-215	Mianus	700
A-217	Brigh	750
A-222	Redwood	700
A-305	Round Hill	350
A-306	Downtown	800
A-307	Perryridge	900
NULL	*NULL*	*NULL*

select *
from *depositor* inner join *account*
on *depositor*.accountnumber=*account*.accountnumber

select *
from *account* inner join *depositor*
on *account*.accountnumber=*depositor*.accountnumber

YEWEN.DBS实验...Query1.sql* | 表 - dbo.account | 表 - d

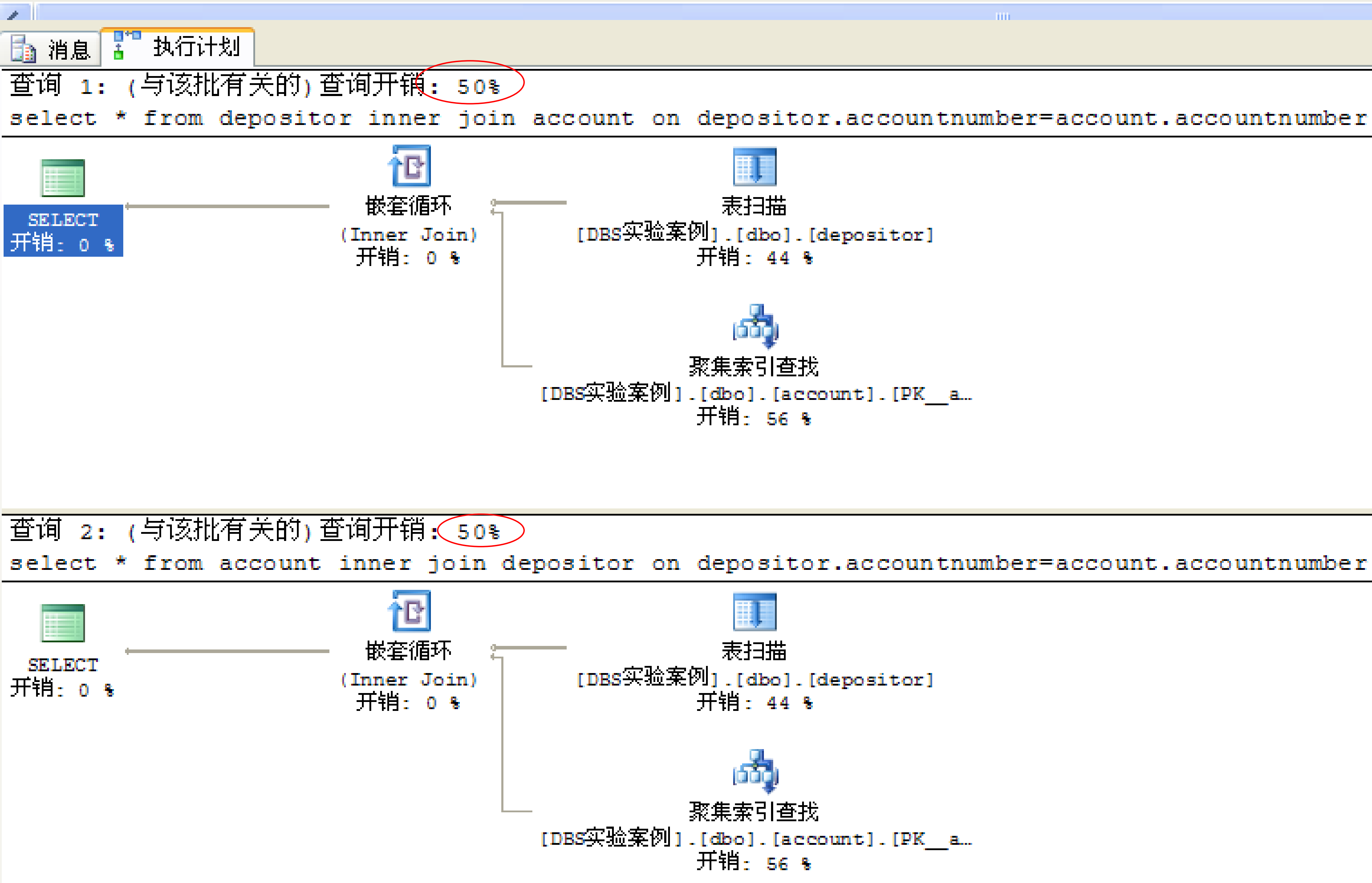
```sql
select *
from depositor inner join account
      on depositor.accountnumber=account.accountnumber

select *
from account inner join depositor
      on depositor.accountnumber=account.accountnumber
```

结果 | 消息

	customername	accountnumber	accountnumber	branchname	balance
1	Hayes	A-102	A-102	Perryridge	400
2	Johnson	A-101	A-101	Downtown	500
3	Johnson	A-201	A-201	Brigh	900
4	Jones	A-217	A-217	Brigh	750
5	Lindsay	A-222	A-222	Redwood	700
6	Simith	A-215	A-215	Mianus	700
7	Turner	A-305	A-305	Round Hill	350

	accountnumber	branchname	balance	customername	accountnumber
1	A-102	Perryridge	400	Hayes	A-102
2	A-101	Downtown	500	Johnson	A-101
3	A-201	Brigh	900	Johnson	A-201
4	A-217	Brigh	750	Jones	A-217
5	A-222	Redwood	700	Lindsay	A-222
6	A-215	Mianus	700	Simith	A-215
7	A-305	Round Hill	350	Turner	A-305



SQL Server查询优化器自动选择元组数少的depositor作为连接操作的outer关系，两条语句的查询执行计划、执行成本一样！！！

Innner Join in SQL Server

```
USE AdventureWorks;
GO
SELECT *
FROM HumanResources.Employee AS e
    INNER JOIN Person.Contact AS c
    ON e.ContactID = c.ContactID
ORDER BY c.LastName
```

此内部联接称为同等联接。它返回两个表中的所有列，但只返回在联接列中具有相等值的行。

```
USE AdventureWorks;
GO
SELECT DISTINCT p.ProductID, p.Name, p.ListPrice, sd.UnitPrice AS 'Selling Price'
FROM Sales.SalesOrderDetail AS sd
    JOIN Production.Product AS p
    ON sd.ProductID = p.ProductID AND sd.UnitPrice < p.ListPrice
WHERE p.ProductID = 718;
GO
```

Equivalence Rules (cont.)

- Rule6. (连接操作的结合率, associative)

a. natural join operations are associative

$$(E1 \bowtie E2) \bowtie E3 = E1 \bowtie (E2 \bowtie E3)$$

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

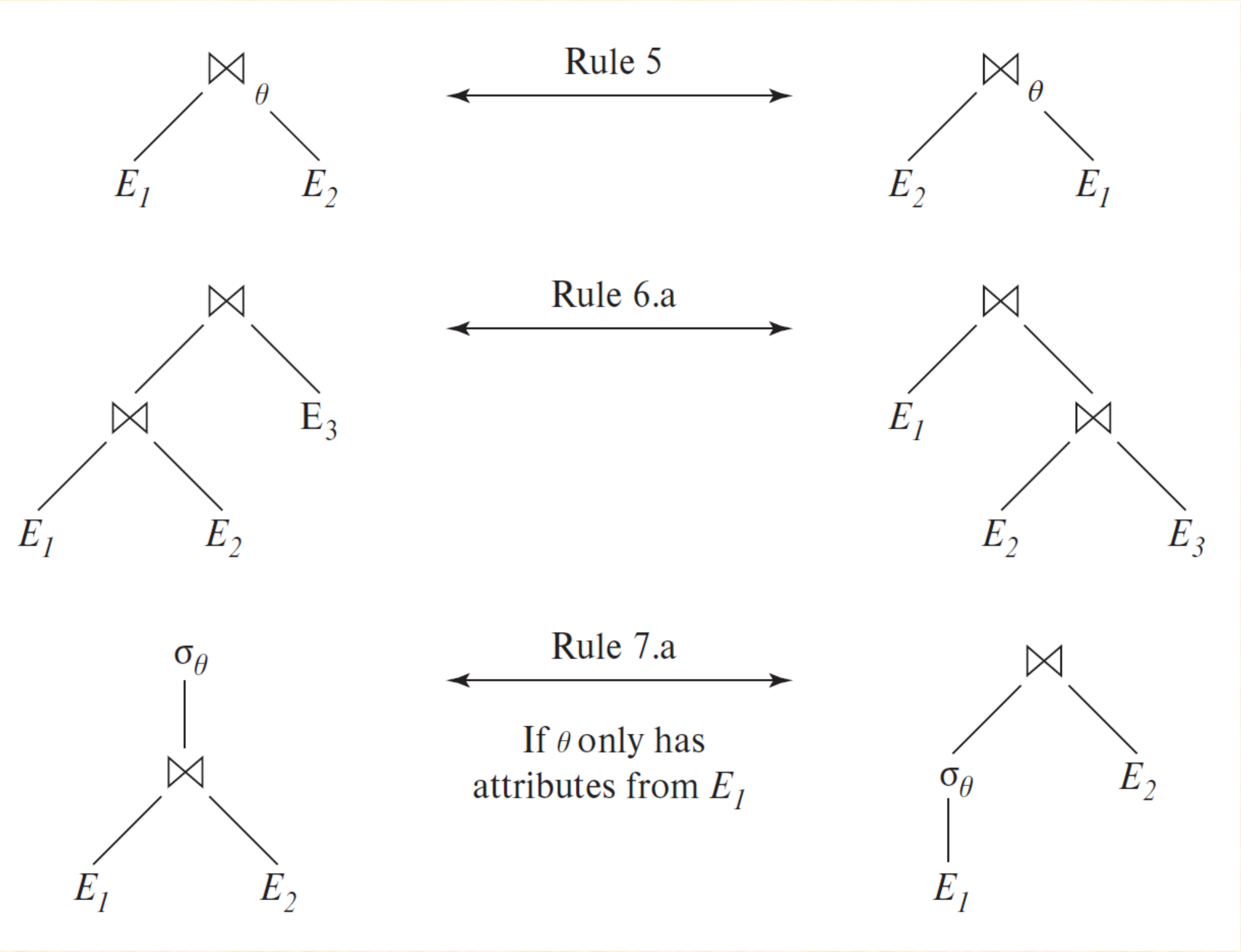


Fig.16.3 Pictorial Depiction of Equivalence Rules

Equivalence Rules (cont.)

- Rule7. Selection operation distributes over theta-join（选择操作对于连接操作的分配率, 选择条件下移）
  - a. when all the attributes in θ_0 involve only the attributes of one of the expressions , e.g. 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

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

  - e.g. in Fig.16.0.2

- When Rule7 is applied, the size of relations participating in theta-join operation is reduced, thus evaluation costs are reduced

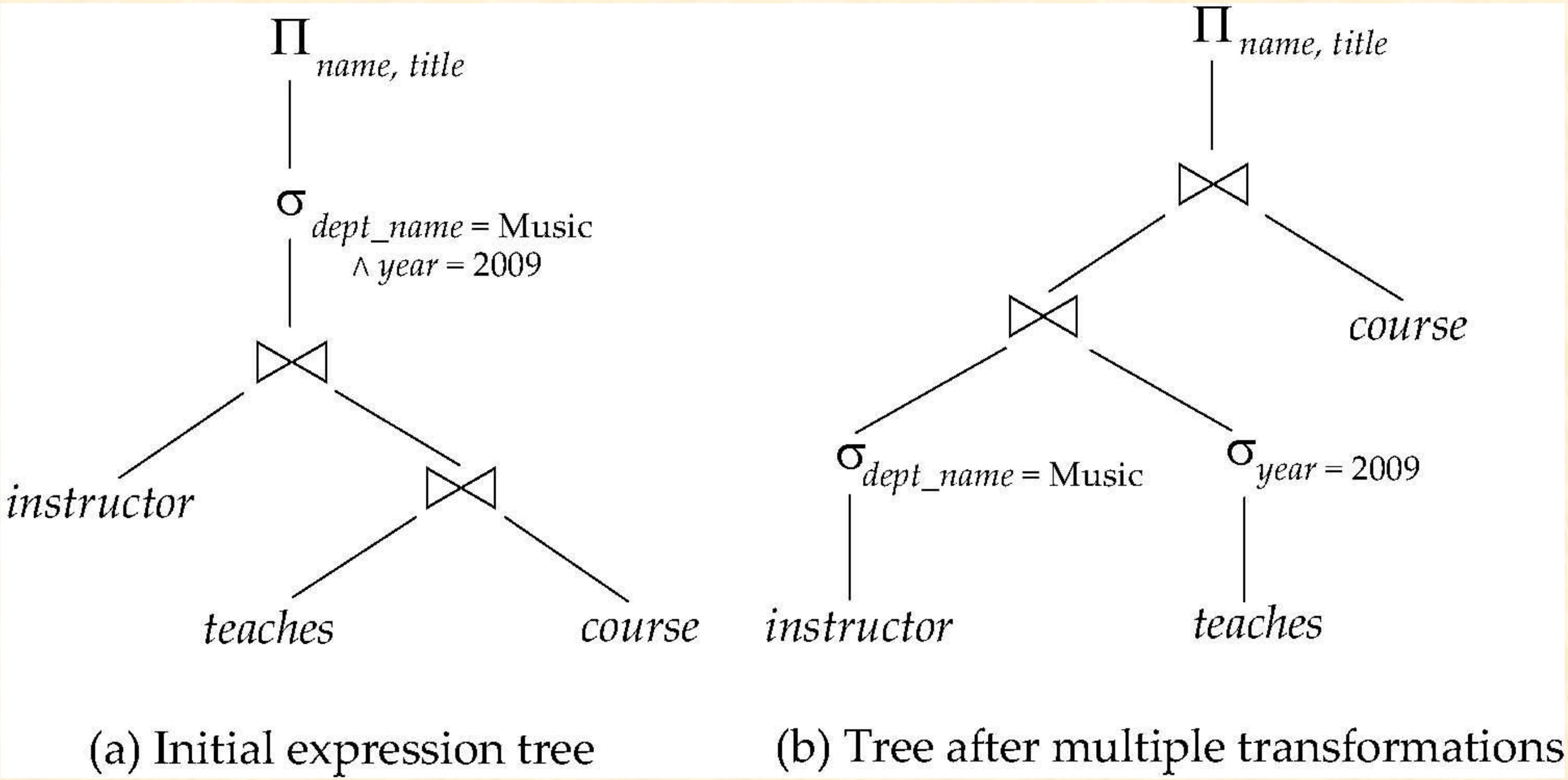


Fig.16.0.2 An example of Rule7

Equivalence Rules (cont.)

- Rule 8. Projection operation distributes over theta-join（投影操作对于连接操作的分配率, 投影下移）
  - a. L_1 and L_2 are the attributes of E_1 and E_2 respectively. if *join condition* θ involves only attributes in $L_1 \cup L_2$, then

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

E1:

instructor (ID, name, dept-name, salary)

E2:

department(dept-name, building, budget)

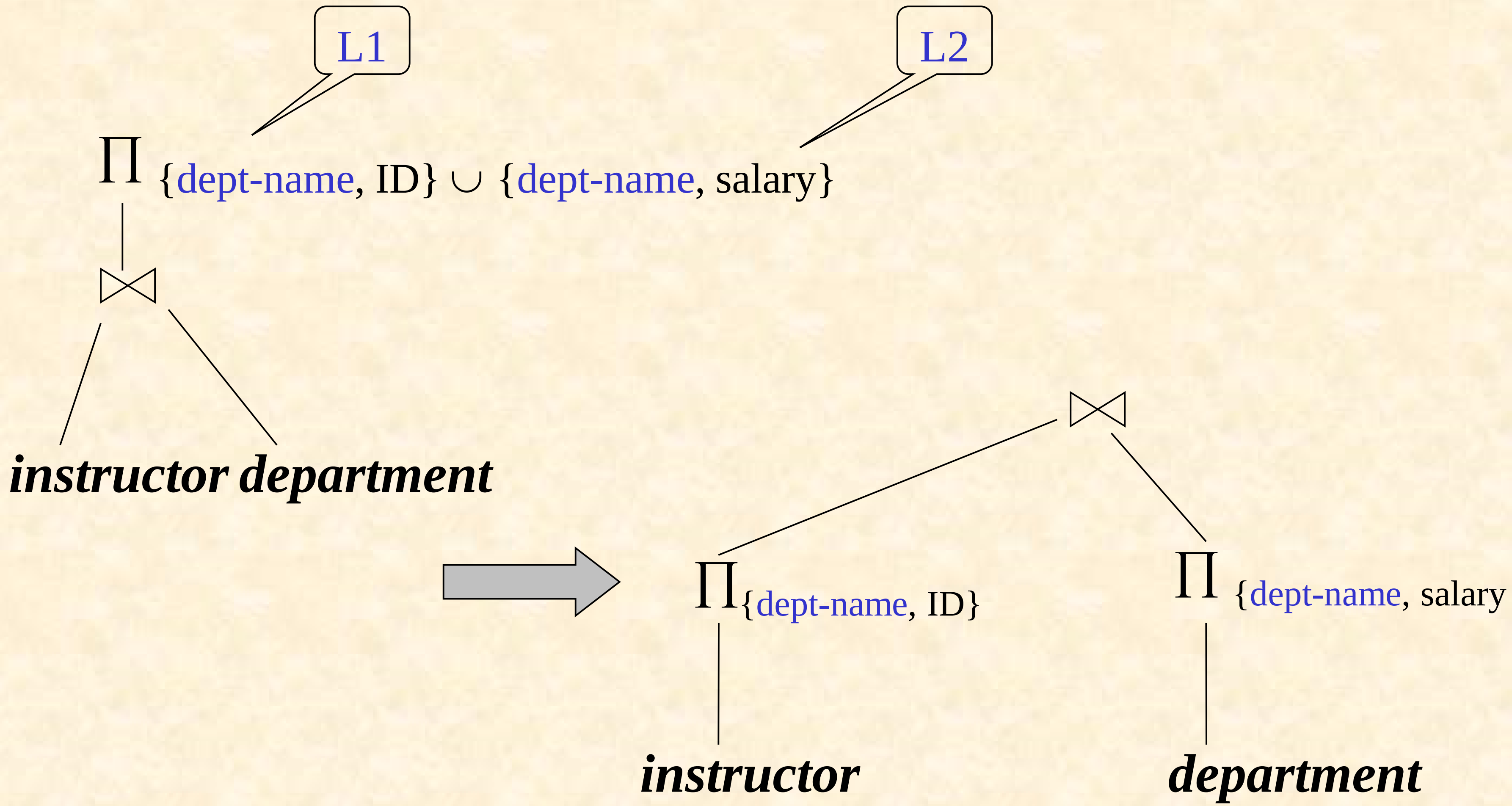
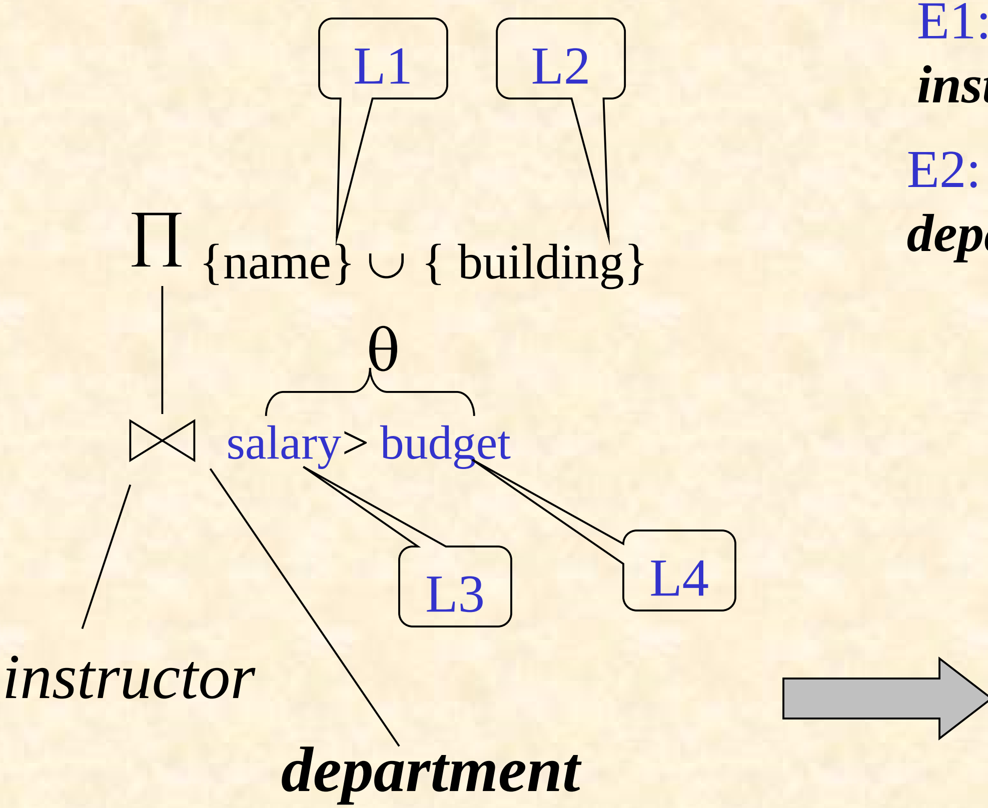


Fig.13.0.3 An example of Rule 8a

Equivalence Rules (cont.)

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 $L_1 \cup L_2$.

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

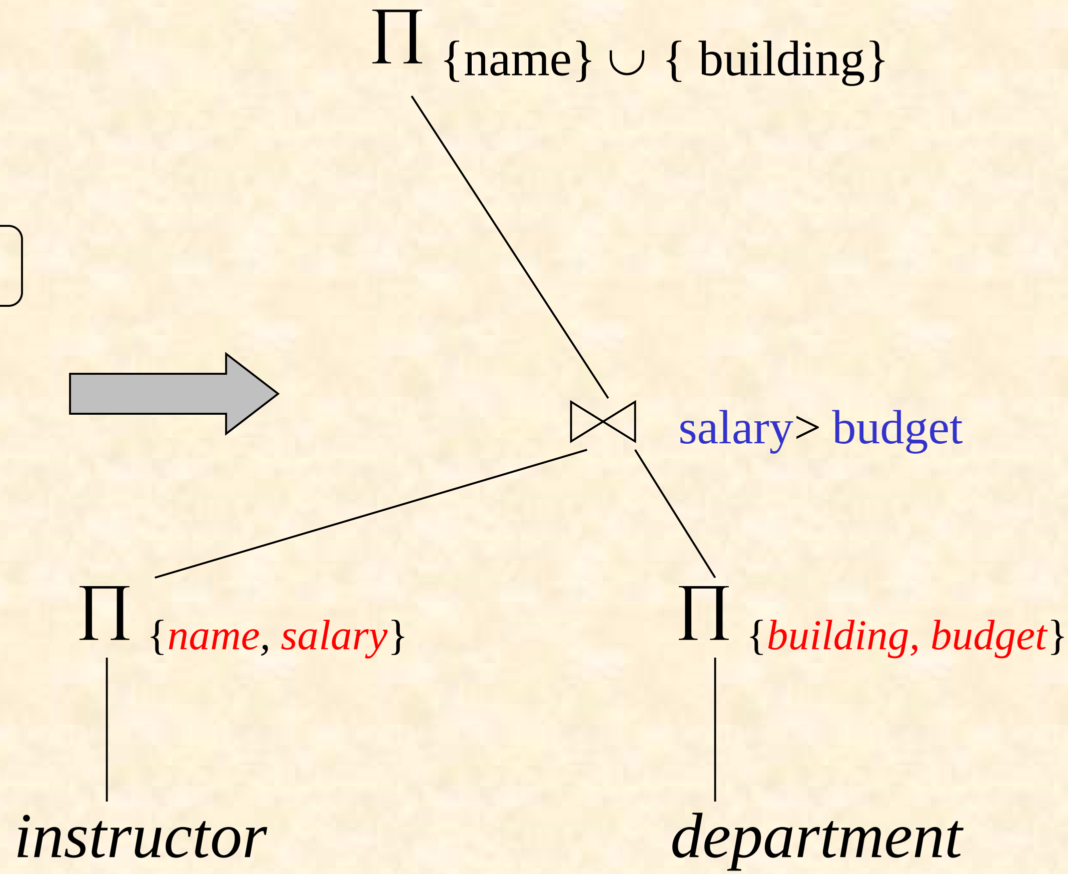


E1:

instructor (ID, name, dept-name, salary)

E2:

department(dept-name, building, budget)



只选后续用到的属性！

Fig.16.0.4 An example of Rule 8b

Equivalence Rules (cont.)

- Rule 9. The set operations *union* and *intersection* are commutative 集合并和交满足交换律

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

  set difference is not commutative

- Rule 10. Set *union* and *intersection* are associative（结合律）

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

Equivalence Rules (cont.)

- Rule11. (选择操作对于集合并、交、差的分配率) The selection operation distributes over ∪, ∩ and –.
  - $\sigma_\theta (E_1 - E_2) = \sigma_\theta (E_1) - \sigma_\theta(E_2)$
    - similarly for ∪ and ∩ in place of –
  - $\sigma_\theta (E_1 - E_2) = \sigma_\theta(E_1) - E_2$
    - similarly for ∩ in place of –, but not for ∪

- Rule12. (投影操作对于集合并的分配率) The projection operation distributes over union

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

北京邮电大学

Example One: Pushing Selections

- Query: Find the *names* of all instructors in the *Music* department, along with the titles of the courses that they teach
  - $\Pi_{name,\ title}(\sigma_{dept\_name=\text{"Music"}}(instructor \bowtie (teaches \bowtie \Pi_{course\_id,\ title}(course))))$
- Transformation using rule 7a.
  - $\Pi_{name,\ title}((\sigma_{dept\_name=\text{"Music"}}(instructor)) \bowtie (teaches \bowtie \Pi_{course\_id,\ title}(course)))$
- Performing the selection as early as possible reduces the size of the relation to be joined.

Example Two： Multiple Transformations

- Find the names of all instructors in the Music department who have taught a course in 2009, along with the titles of the courses that they taught
  - $\Pi_{name,\ title}(\sigma_{dept\_name=\ \text{"Music"} \wedge year\ =\ 2009}$
    $(instructor \bowtie (teaches \bowtie \Pi_{course\_id,\ title}(course))))$
- Transformation using join associatively (Rule 6a):
  - $\Pi_{name,\ title}(\sigma_{dept\_name=\ \text{"Music"} \wedge year\ =\ 2009}$
    $((instructor \bowtie teaches) \bowtie \Pi_{course\_id,\ title}(course)))$
- Second form provides an opportunity to apply the "perform selections early" rule, resulting in the subexpression

$$\sigma_{dept\_name\ =\ \text{"Music"}}(instructor) \bowtie \sigma_{year\ =\ 2009}(teaches)$$

Example Three: Pushing Projections

- Consider: $\Pi_{name,\ title}(\sigma_{dept\_name=\text{“Music”}}(instructor) \bowtie teaches) \bowtie \Pi_{course\_id,\ title}(course))))$
- When we compute

  $(\sigma_{dept\_name = \text{“Music”}}(instructor) \bowtie teaches)$

  we obtain a relation whose schema is:
  (*ID*, *name*, *dept_name*, *salary*, *course_id*, *sec_id*, *semester*, *year*)
- Push projections using equivalence rules 8a and 8b; eliminate unneeded attributes from intermediate results to get:

  $$\Pi_{name,\ title}(\Pi_{name,\ course\_id}(\sigma_{dept\_name=\text{“Music”}}(instructor) \bowtie teaches)) \bowtie \Pi_{course\_id,\ title}(course))))$$
- Performing the projection as early as possible reduces the size of the relation to be joined.

16.2.3 Join Ordering（连接次序）

- For all relations r_1, r_2, and r_3,

$$(r_1 \bowtie r_2) \bowtie r_3 = r_1 \bowtie (r_2 \bowtie r_3)$$

- If $r_2 \bowtie r_3$ is quite large and $r_1 \bowtie r_2$ is small, we choose

$$(r_1 \bowtie r_2) \bowtie r_3$$

so that we compute and store a smaller temporary relation.

Example

- Consider the expression

$$\Pi_{name,\ title}(\underline{\sigma_{dept\_name=\text{"Music"}}(instructor)} \bowtie \underline{teaches} \bowtie \underline{\Pi_{course\_id,\ title}(course)})$$

- Could compute $teaches \bowtie \Pi_{course\_id,\ title}(course)$ first, and join result with

  $$\sigma_{dept\_name=\text{"Music"}}(instructor)$$

  but the result of the first join is likely to be a large relation.
- Only a small fraction of the university's instructors are likely to be from the *Music* department
  - it is better to compute

    $$\underline{\sigma_{dept\_name=\text{"Music"}}(instructor) \bowtie teaches}$$

    first.

略 § 16.3 Estimating Statistics of Expression Results

- How to estimate the cost of an ***operation p***
  - estimating the size of the resultant relations produced by the operation ***p***, according to statistics in the ***catalog*** about the relations on which the operation is conducted
    - e.g. $\Pi_{\text{customer-name}}(\sigma_{\text{balance<2500}}(account) \bowtie customer)$
  - computing the cost of the operation in terms of *disk access*, by means of the cost formulae shown in chapter15
    - disk access is the predominant cost, and the *number of block transfers from disk* is used as a measure of the actual cost of evaluation.

16.3.1 Statistics in Catalog for Optimization

- n_r: number of tuples in a relation r
- b_r: number of blocks containing tuples of r
- l_r: size of a tuple of r in bytes
- f_r: blocking factor of r
  - the number of tuples of r that fit into one block

- If tuples of r are stored together physically in a file, then:

$$b_r = \left\lceil \frac{n_r}{f_r} \right\rceil$$

Statistics in Catalog for Optimization (cont.)

- *V(A, r):* number of distinct values that appear in *r* for attribute *A*
  - same as the size of $\prod_A(r)$, $|\prod_A(r)|$

- *Statistics about indices*, such as heights of B^+-trees, that is HT_i, and the number of leaf pages in the indices, are also in catalog

Statistics in Catalog for Optimization (cont.)

- An example for statistical information
  - $\boldsymbol{n_{account}}$ = 10000 (*account* has 10,000 tuples)
  - $\boldsymbol{f_{account}}$ = 20 (20 tuples of *account* fit in one block)
  - *V(**branch-name, account**)* = 50 (50 *branches*)
  - *V(**balance, account**) = 500* (500 different *balance* values)
  - assume the following indices exist on account
    - a clustered/primary, B^+-tree index for the attribute ***branch-name***
    - a secondary, B^+-tree index for the attribute ***balance***

16.3.2 ***Selection*** Size Estimation

- The size estimation of the result of a ***selection*** operation depends on the *selection predicates*, that is *single equality predicate*, *single comparison predicate*, and *combination of predicates*

- Equality selection $\boldsymbol{\sigma_{A=a}(r)}$
  - assuming uniform distribution of values in relations, and the value ***a*** appears in attribute ***A*** of some record in ***r***
  - the selection is estimated to have
    - n_r / V(A, r) tuples satisfying ***A***=a
    - $\lceil n_r / V(A, r) / f_r \rceil$ blocks that these tuples occupy

Selection Size Estimation (cont.)

- Comparison selections $\sigma_{A\leq v}(r)$
  - the lowest and highest values of attributes, that is min(A, r) and max(A, r), are available in catalog
  - the estimated number of records/tuples satisfying comparison condition $A \leq V$
    - 0, if $v <$ min(A, r)
    - n_r, if $v \geq$ max(A, r)
    - $$n_r \cdot \frac{v - \min(A, r)}{\max(A, r) - \min(A, r)}$$, otherwise
  - in absence of statistical information, the *cost* is assumed to be $n_r / 2$

Selection Size Estimation (cont.)

- The **selectivity** of a condition θ_i is the ***probability*** that a tuple in the relation r satisfies θ_i . If s_i is the number of satisfying tuples in r, the selectivity of θ_i is given by s_i / n_r.
- **Conjunction:** $\sigma_{\theta 1 \wedge \theta 2 \wedge \ldots \wedge \theta n}(r)$. The estimate for number of tuples in the result is: $$n_r * \frac{s_1 * s_2 * \ldots * s_n}{n_r^n}$$
- **Disjunction:** $\sigma_{\theta 1 \vee \theta 2 \vee \ldots \vee \theta n}(r)$. Estimated number of tuples:
$$n_r * \left(1 - (1 - \frac{s_1}{n_r}) * (1 - \frac{s_2}{n_r}) * \ldots * (1 - \frac{s_n}{n_r})\right)$$
- **Negation:** $\sigma_{\neg\theta}(r)$. Estimated number of tuples:
$$n_r - size(\sigma_\theta(r))$$