第5讲分布式数据库中的查询处理和优化

1. 分布式查询优化概述
2. 分布式查询优化基础知识
3. 分布式查询分类和层次结构
4. 基于关系代数等价变换的查询优化处理
5. 基于半连接算法的查询优化处理
6. 基于直接连接算法的查询优化处理
7. 直接连接操作的常用策略

1 分布式查询优化概述

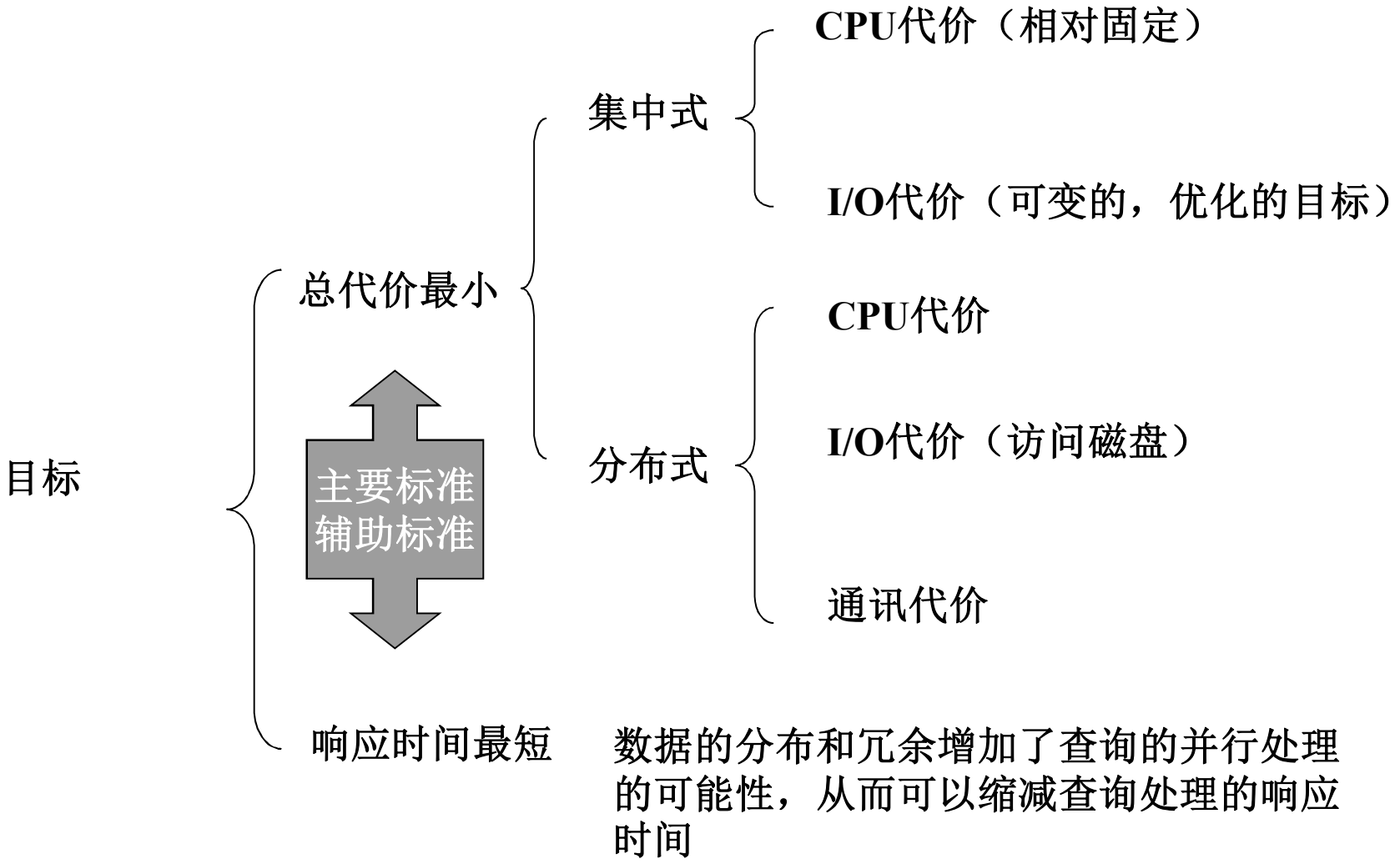
1.1 分布式查询优化的目标

查询处理问题

- 集中式
  - 查询转换为代数表达式
  - 从所有等价表达式中选择最优的代数表达式
- 分布式
  - 除了集中式问题外，还有
  - 站点之间交换数据的操作
  - 选择最优的执行站点(分布）
  - 数据被传送的方式

1 分布式查询优化概述

1.1 分布式查询优化的目标



1 分布式查询优化概述

1.2 分布式查询优化准则和代价分析

准则：

使得通讯费用最低和响应时间最短，即以最小的总代价，在最短的响应时间内获得需要的数据。

1. 通讯费用与所传输的数据量和通信次数有关
2. 响应时间和通信时间有关，也与局部处理时间有关

查询代价分析

1. 远程通讯网络

局部处理时间可以忽略不计，减少通讯代价是主要目标

2. 高速局域网

传输时间比局部处理时间要短很多，以响应时间作为优化目标，局部处理时间是关键

1.3 分布式查询策略的重要性

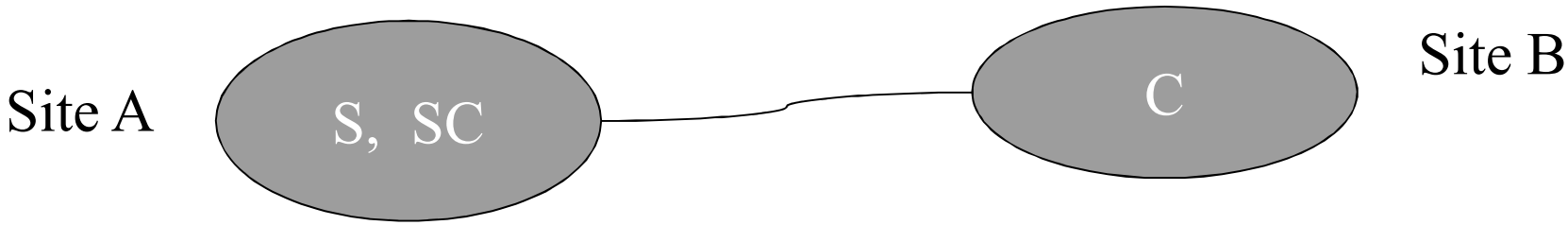
- 例子

S(s#, sname, age, sex)　　10^4 元组　Site A

C(c#, cname, teacher)　　10^5 元组　Site B

SC(s#, c#, grade)　　10^6 元组　Site A

每个元组长度100bit, 通讯传输速度 10^4 bit/sec, 通讯延迟 1sec



1.3 分布式查询策略的重要性

查询: 所有选修maths 课的男生学号和姓名.

```
SELECT   s#, sname
FROM   S, C, SC
WHERE  S.s#=SC.s#  AND
              C.c#=SC.c#  AND
              sex='男'  AND cname='maths';
```

1.3 分布式查询策略的重要性

- 代价公式

  QC = I/O 代价 + CPU 代价 + 通讯代价

- 通讯代价

  TC = 传输延迟时间C0

  + (传输数据量X /数据传输速率C1)

1 分布式查询优化概述

1.3 分布式查询策略的重要性

四种查询策略

策略1:

A 传C B

S,SC 通信1次 C

把关系C传输到A地,在A地处理查询

T1 = 1 + (10^5 * 100 / 10^4)

≈ 10^3 秒 ≈ 16.7分钟

策略2:

A 传S,SC B

S,SC 通信2次 C

把关系S和SC传输到B地,在B地处理查询

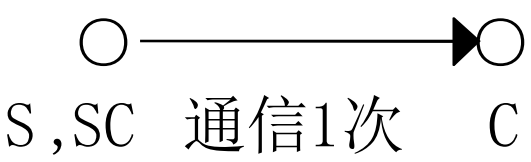
T2 = 2+ (10^4+10^6) * 100 / 10^4

≈10100秒≈ 2.8小时

1.3 分布式查询策略的重要性

四种查询策略

策略 3：

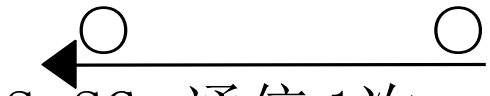


先在A地求出男生选课元组，有10^5 个
再把结果传输到B地，在B地执行查询，

$T5 = 1 + (10^5 * 100) / 10^4$

≈ 1000 秒 ≈ 16.7 分

策略 4：



先在B地求出为‘ MATHS ’的元组，有10个
再把结果传输到 A 地 ,在A 地执行查询，

$T6 = 1 + (10 * 100) / 10^4$ ≈ 1秒

2 分布式查询优化中的基础知识

2.1 关系代数知识回顾

关系代数

——是根据查询来生成新表的方法的集合

关系代数的运算分成两种类型：

- 集合运算，表实际上是“行”的集合，集合的运算是只涉及“行”的运算；
- 专门的关系运算，既涉及到“行”，也涉及到“列”的运算。

2 分布式查询优化中的基础知识

2.1 关系代数知识回顾

关系模型

集合运算

名称	符号	示例
并	∪	**R ∪ S，或 R UNION S**
交	∩	**R ∩ S，或 R INTERSECT S**
差	–	**R – S， 或 R MINUS S**
笛卡儿积	×	**R × S， 或 R TIMES S**

专门的关系运算

名称	符号	示例
投影	π	$\pi_{Ai1\ldots Aik}R$
选择	σ	$\sigma_{C\#='C2'}R$
连接	⋈	**R ⋈ S，或 R JOIN S**
除	÷	**R ÷ S， 或 R DIVIDEBY S**

2 分布式查询优化中的基础知识

2.1 关系代数知识回顾

关系模型

其它关系运算

名称	符号	键盘格式	示例
外连接	$\bowtie_O$	**OUTERJ**	**R** $\bowtie_O$ **S，或 R OUTERJ S**
左外连接	$\bowtie_{LO}$	**LOUTERJ**	**R** $\bowtie_{LO}$ **S，或 R LOUTERJ S**
右外连接	$\bowtie_{RO}$	**ROUTERJ**	**R** $\bowtie_{RO}$ **S， 或 R ROUTERJ S**
θ连接	$\bowtie_{Ai\ \theta\ Bj}$	**JN(Ai θ Bj)**	**R** $\bowtie_{Ai\ \theta\ Bj}$ **S，或 R JN(Ai θ Bj) S**

这里，θ是比较运算符，可以是>, <, >=, <=, =, <>

2 分布式查询优化中的基础知识

2.1 关系代数知识回顾

- 并、交、 差运算

R1

A	B	C
a1	b1	c1
a1	b2	c2
a2	b2	c1

R2

A	B	C
a1	b2	c2
a1	b3	c2
a2	b2	c1

R1 ∪ R2

A	B	C
a1	b1	c1
a1	b2	c2
a1	b3	c2
a2	b2	c1

2 分布式查询优化中的基础知识

2.1 关系代数知识回顾

广义笛卡尔积

两个分别为n目和m目的关系R和S的广义笛卡尔积是一个(n+m)列的元组的集合。元组的前n列是关系R的一个元组，后m列是关系S的一个元组。若R有k1个元组，S有k2个元组，则关系R和关系S的广义笛卡尔积有k1×k2个元组。记作：

$$R \times S = \{ \overbrace{t_r \; t_s} \mid t_r \in R \wedge t_s \in S \}$$

R1

A	B	C
a1	b1	c1
a1	b2	c2
a2	b2	c1

R2

A	B	C
a1	b2	c2
a1	b3	c2
a2	b2	c1

R1×R2

A	B	C	A	B	C
a1	b1	c1	a1	b2	c2
a1	b1	c1	a1	b3	c2
a1	b1	c1	a2	b2	c1
a1	b2	c2	a1	b2	c2
a1	b2	c2	a1	b3	c2
..	..	..	..	..	..

连接运算（θ 连接）

连接运算是从两个关系的笛卡尔积中选取属性间满足一定条件的元组。记做：$R \underset{F}{\infty} S$. 其中，**F**是条件表达式，它涉及到对两个关系中的属性的比较。

$$R \underset{A\,\theta\,B}{\bowtie} S = \{ \widehat{t_r\ t_s} \mid t_r \in R \wedge t_s \in S \wedge t_r[A]\ \theta\ t_s[B] \}$$

例　设关系**R**、**S**如下图：　　$R \underset{C<E}{\infty} S$

R

A	B	C
a1	b1	5
a1	b2	6
a2	b3	8
a2	b4	12

S

B	E
b1	3
b2	7
b3	10
b3	2
b5	2

$R \underset{C<E}{\infty} S$

A	R.B	C	S.B	E
a1	b1	5	b2	7
a1	b1	5	b3	10
a1	b2	6	b2	7
a1	b2	6	b3	10
a2	b3	8	b3	10

自然连接

另一种是自然连接。自然连接是一种特殊的等值连接，它要求两个关系中进行比较的分量必须是相同的属性组，并且要在结果中把重复的属性去掉。 $R \bowtie S=\{\overbrace{t_r\ t_s}[\overline{B}] \mid t_r \in R \wedge t_s \in S \wedge t_r[B]=t_s[B]\}$

例6　关系**R**、**S**的自然连结：R ∞ S

R ∞ S

A	B	C	E
a1	b1	5	3
a1	b2	6	7
a2	b3	8	10
a2	b3	8	2

$R \underset{R.B=S.B}{\infty} S$

A	R.B	C	S.B	E
a1	b1	5	b1	3
a1	b2	6	b2	7
a2	b3	8	b3	10
a2	b3	8	b3	2

半连接

在R、S自然连接后仅保留对R的属性的投影，记为：R ∝ S

例　关系R、S的半连接：

R

A	B	C
a1	b1	5
a1	b2	6
a2	b3	8
a2	b4	12

S

B	E
b1	3
b2	7
b3	10
b3	2
b5	2

R ∞ S

A	B	C	E
a1	b1	5	3
a1	b2	6	7
a2	b3	8	10
a2	b3	8	2

R ∝ S

A	B	C	
a1	b1	5	
a1	b2	6	
a2	b3	8	

关系代数表达式

设教学数据库中有三个关系：

学生关系S（S#，SNAME，SD， AGE ）

课程关系C（C#，CNAME, TEACHER）

学习关系SC（S#，C#，GRADE）

例　检索学习课程号为C2的学生学号与成绩

$\sigma_{C\#='C2'}(SC)$

学号 S#	课程号 C#	学习成绩 GRADE
S1	C2	A
S2	C2	C
S3	C2	B
..	..	..

$$\pi_{S\#,GRADE}(\sigma_{C\#='C2'}(SC))$$

SC

学号 S#	课程号 C#	学习成绩 GRADE
S1	C1	A
S1	C2	A
S1	C3	A
S1	C5	B
S2	C1	B
S2	C2	C
S2	C4	C
S3	C2	B
..	..	..

例 **检索学习课程号为C2的学生学号和姓名**

S

学号 S#	学生姓名 SNAME	所属系名 SD	学生年龄 SA
S1	A	CS	20
S2	B	CS	21
S3	C	MA	19
S4	D	CI	19
S5	E	MA	20
..	..	..	..

SC

学号 S#	课程号 C#	学习成绩 GRADE
S1	C1	A
S1	C2	A
S1	C3	A
S1	C5	B
S2	C1	B
S2	C2	C
..	..	..

S ∞ SC

学号 S#	学生姓名 SNAME	所属系名 SD	学生年龄 SA	课程号 C#	学习成绩 GRADE
S1	A	CS	20	C1	A
S1	A	CS	20	C2	A
S1	A	CS	20	C3	A
S1	A	CS	20	C5	A
S2	B	CS	21	C1	B
S2	B	CS	21	C2	C
..	..	..	..	..	..

$\sigma_{C\#='C2'}(S \infty SC)$

S#	SNAME
S1	A
S2	B

$\pi_{S\#,SNAME}(\sigma_{C\#='C2'}(S \infty SC)) = \pi_{S\#,SNAME}(S \infty_{C\#='C2'} SC)$

例 **检索学习课程号为C2或C3的学生学号和所在系**

S

学号 S#	学生姓名 SNAME	所属系名 SD	学生年龄 SA
S1	A	CS	20
S2	B	CS	21
S3	C	MA	19
S4	D	CI	19
S5	E	MA	20
..	..	..	..

SC

学号 S#	课程号 C#	学习成绩 GRADE
S1	C1	A
S1	C2	A
S1	C3	A
S1	C5	B
S2	C1	B
S2	C2	C
..	..	..

$$\pi_{S\#,SD}(S \quad \infty \pi_{S\#}(\sigma_{C\#='C2' \vee C\#='C3'}(SC)))$$

2 分布式查询优化中的基础知识

2.1 用关系代数和SQL语句表示一个查询

关系代数基本操作：

并（∪）、交（∩）、笛卡尔积（ ×）、选择（σ）、投影（π）

关系代数导出操作：

差（－）、除（÷）、 θ 连接（∞ θ ）、自然连接（∞）、半连接（∝）

- SQL与代数的等价描述
- E1
  - SELECT sname FROM S, SC
    WHERE S.s#=SC.s# and SC.c#='c03';
  - 代数描述

$$\pi_{sname}(\sigma_{s.s\#=SC.s\# \text{ and } SC.c\#='c03'}(S \times SC))$$

- E2
  - SELECT sname FROM S WHERE S.s# in
    (SELECT SC.s#
    FROM SC WHER c#='c03');
  - 代数描述

$$\pi_{sname}(\sigma_{s.s\#=SC.s\#}(S \times \sigma_{SC.c\#='c03'}, SC))$$

- E3
  - SELECT sname FROM S , (SELECT SC.s#
    FROM SC WHER c#='c03') SCC
    WHERE S.s# = SCC.s# ;
  - 代数描述

$$\pi_{sname}(S \infty \sigma_{SC.c\#='c03'}, SC)$$

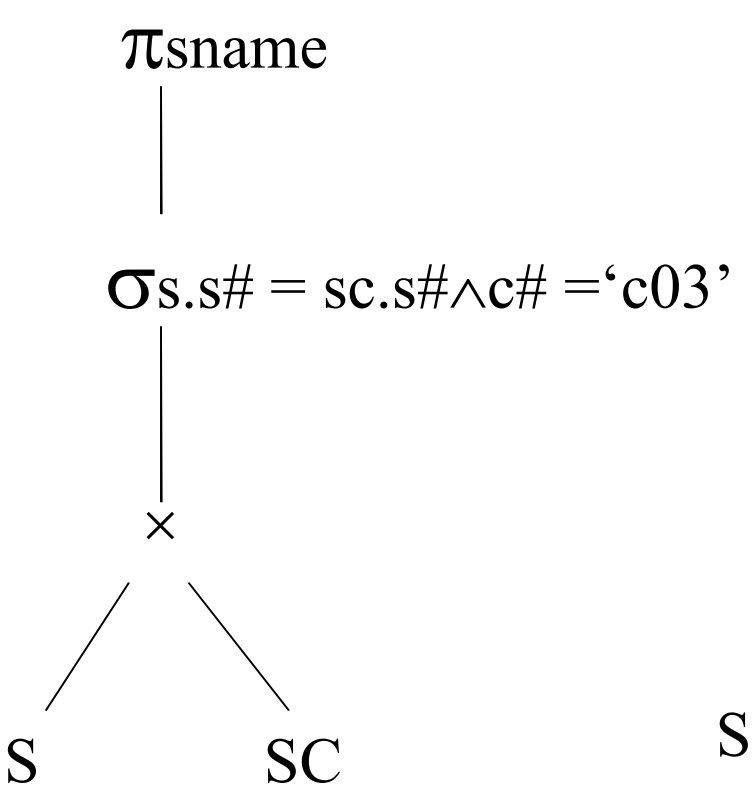
2 分布式查询优化中的基础知识

2.2 查询树

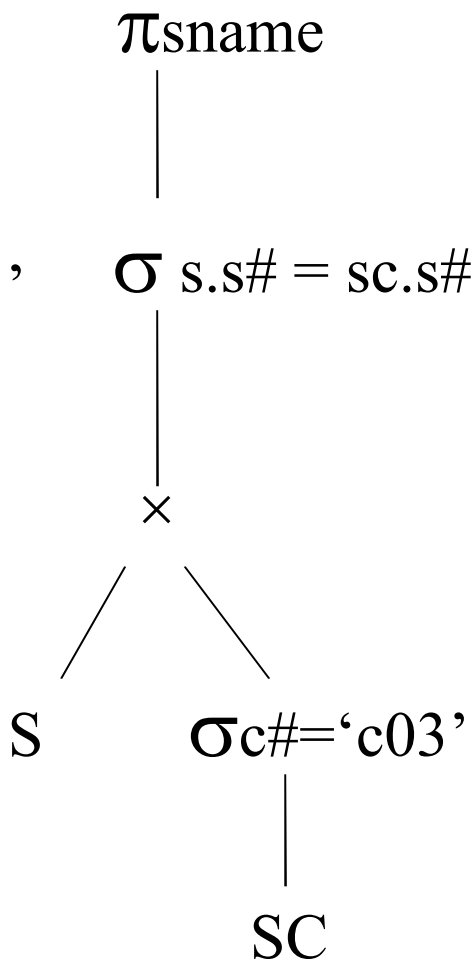
叶子表示已知关系

节点表示一个一元或二元操作符

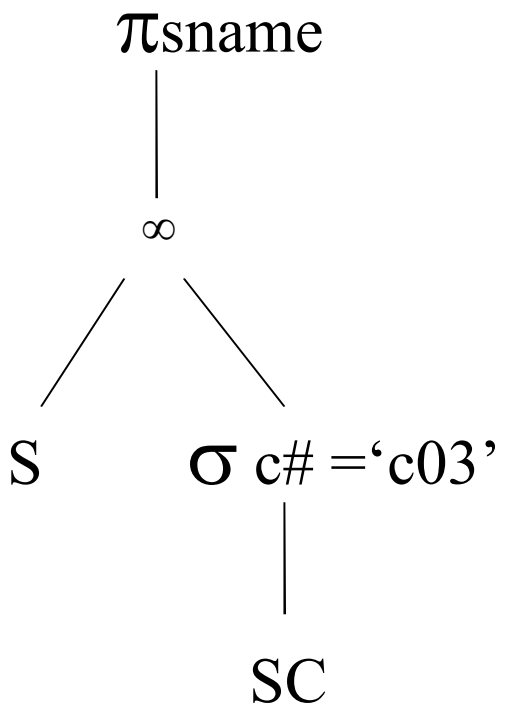
树根表示查询结果



(a)对于E1的查询树



(b)对于E2的查询树



(c)对于E3的查询树

2 分布式查询优化中的基础知识

2.3 等价变换规则的概念和术语

- 一元操作：只涉及一个操作对象

  σ (SL), π (PJ)

- 二元操作：涉及两个操作对象

  $\cup$ (并), $\cap$（交）, $-$ (差), $\times$ (笛卡尔积),

  $\infty\theta$（ θ 连接），∞ (自然连接), $\propto$ (半连接), $\div$（除）

2 分布式查询优化中的基础知识

2.3 等价变换规则的概念和术语

- 空值的变换　　等价变换规则

$R \cup \varnothing = R$　$R \cap \varnothing = \varnothing$　$R \infty \varnothing = \varnothing$　$\varnothing \propto R = \varnothing$

$\varnothing - R = \varnothing$　$R - \varnothing = R$　$R \infty_{\theta} \varnothing = \varnothing$　$R \times \varnothing = \varnothing$

$\sigma(\varnothing) = \varnothing$　$\pi(\varnothing) = \varnothing$

- 自身操作的等价

$R \cup R = R$　$R \cap R = R$　$R \infty R = R$

2 分布式查询优化中的基础知识

2.3 等价变换规则的概念和术语

- 一元操作　　等价变换规则

$\sigma_{F1}(\sigma_{F2}(R)) = \sigma_{F1 \text{and } F2}(R)$

F只涉及$A_1 \ldots A_n$的属性：

$\pi_{A1,\ldots,An}(\sigma_F(E)) = \sigma_F(\pi_{A1,\ldots,An}(E))$

F还涉及不在$A_1 \ldots A_n$中的$B_1 \ldots B_m$的属性：

$\pi_{A1,\ldots,An}(\sigma_F(E)) = \pi_{A1,\ldots,An}(\sigma_F(\pi_{A1,\ldots,An,B1,\ldots,Bm}(E)))$

2 分布式查询优化中的基础知识

2.3 等价变换规则的概念和术语

- 交换率

$O_1(O_2(R)) = O_2(O_1(R))$

条件:

– O_1 O_2 是 选择操作 时总成立

– O_1 O_2 是 投影操作 时要求其属性集合相等

– O_1 与 O_2 是投影和选择操作时:

$\pi_{A1,\ldots An}(\sigma_F(R)) = \sigma_F(\pi_{A1,\ldots An}(R))$ 的条件是 F中的属性是 A1, …. An 的子集

$R \infty S = S \infty R \qquad R \times S = S \times R$

$R \cup S = S \cup R \qquad R \cap S = S \cap R$

$R \propto S \neq S \propto R \qquad R - S \neq S - R$

2 分布式查询优化中的基础知识

2.3 等价变换规则的概念和术语

- 结合率

  R B(S B T) = (R B S) B T

  B: 二元操作

  ∞, ×, ∪, ∩等总成立

2 分布式查询优化中的基础知识

2.3 等价变换规则的概念和术语

- 分配率

$O(R\ B\ S) = O(R)\ B\ O(S)$

O:一元操作 B:二元操作

– $\sigma_F(R \times S)$ 其中F = F1 and F2,
若F1有R属性, F2有S属性, 则

$\sigma_F(R \times S) = \sigma_{F1}(R) \times \sigma_{F2}(S)$

若F1只有R属性, F2有R与S属性, 则

$\sigma_F(R \times S) = \sigma_{F2}(\sigma_{F1}(R) \times S)$

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

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

$\sigma_F(R \infty S) = \sigma_F(R) \infty \sigma_F(S)$

3 分布式查询的分类与层次结构

3.1 分布式查询分类

- 局部查询：只涉及本地. 单个站点的数据, 优化同集中式
  - 选择和投影早做，中间结果大大减少
  - 连接前进行预处理（属性排序、属性索引）
  - 同时执行一串投影和选择操作
  - 找出公共子表达式
- 远程查询：也只涉及单个站点的数据, 但要远程通讯, 选择站点
  - 选择查询应用最近的冗余分配站点
- 全局查询：涉及多个站点数据, 优化复杂

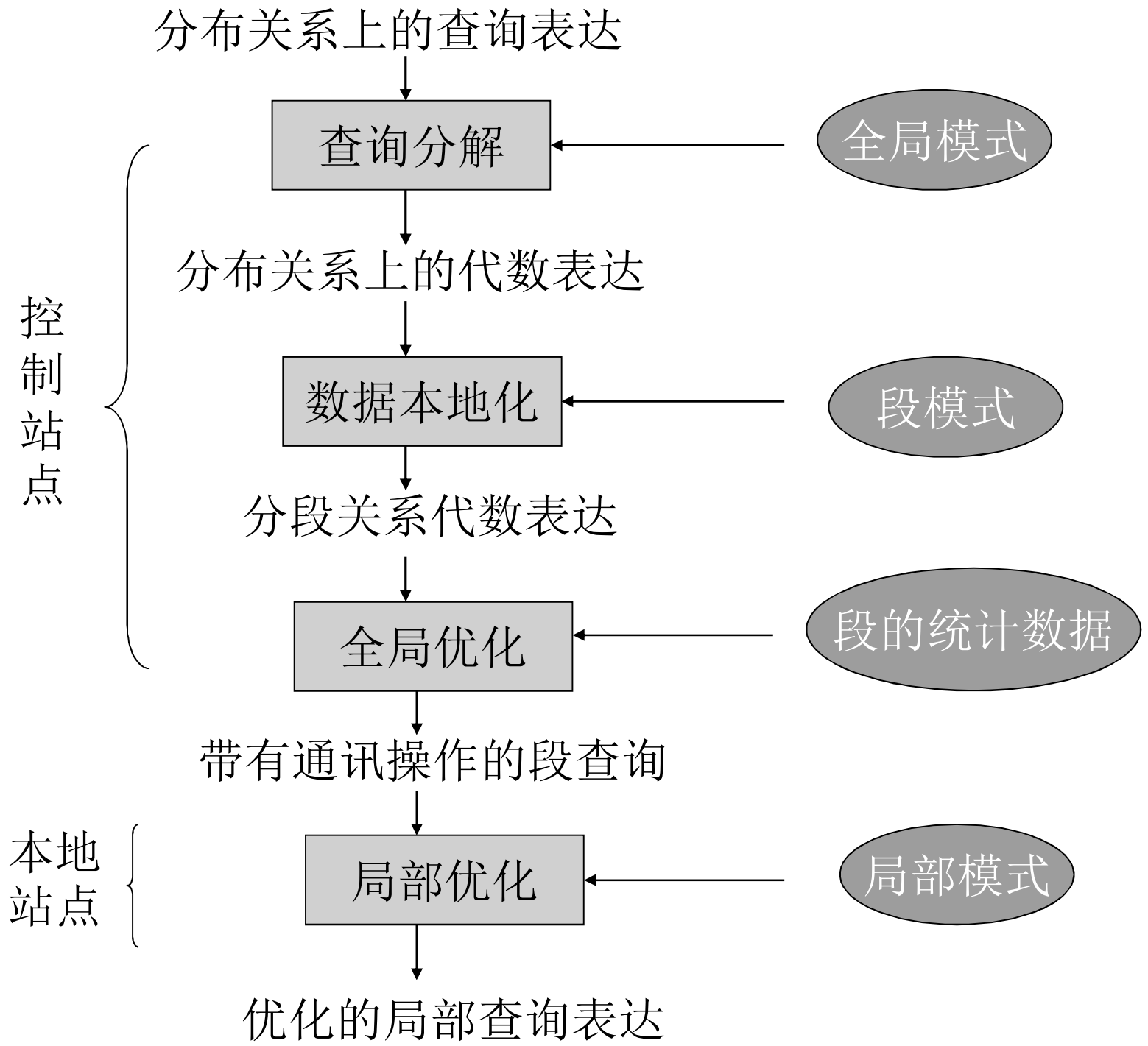
3 分布式查询优化概述

3.1 分布式查询分类

全局查询

- 具体化
  - 对查询进行分解，确定查询使用的物理副本，落实查询对象
  - 非冗余具体化，所有要访问对象只有一个副本
  - 冗余具体化，多个副本，研究如何如何选择副本，使通信代价最小
- 确定操作执行的顺序
  - 确定二元操作连接和并操作的顺序
    - 先执行所有连接操作，再执行并操作
    - 先执行部分并操作，再执行连接操作
  - 选择和投影尽可能早进行
- 确定操作执行的方法
  - 把若干个操作连接起来在一次数据库访问中完成，确定可用的访问路径
  - 连接方法在查询优化中起着重要作用
- 确定执行的站点
  - 执行站点不一定是发出查询的站点
  - 考虑通讯费用和执行效率

分布查询的层次


分布关系上的查询表达
查询分解
全局模式
分布关系上的代数表达
控制站点
数据本地化
段模式
分段关系代数表达
全局优化
段的统计数据
带有通讯操作的段查询
本地站点
局部优化
局部模式
优化的局部查询表达

3 分布式查询优化概述

3.2 分布式查询处理的层次结构

- 查询分解
  - 将查询问题（SQL）转换成一个定义在全局关系上的关系代数表达式
  - 需要从全局概念模式中获得转换所需要的信息
- 数据本地化
  - 具体化全局关系上的查询，落实到合适的片段上的查询
  - 即将全局关系上的关系代数表达式变换为相应片段上的关系代数表达式
- 全局优化
  - 优化目标是寻找一个近于最优的执行策略（操作次序）
  - 输出是一个优化的、片段上的关系代数查询
- 局部优化
  - 输入是局部模式
  - 它由该站点上的DBMS进行优化

4 基于关系代数等价变换的查询优化处理

4.1 基本原理和实现方法

- 基本原理
  1. 查询问题——〉关系代数表达式
  2. 分析得到查询树
  3. 进行全局到片段的变换得到基于片段的查询树
  4. 利用关系代数等价变换规则的优化算法，尽可能先执行选择和投影操作
- 优化算法
  1. 连接和合并尽可能上提（树根方向）
  2. 选择和投影操作尽可能下移（叶子方向）

4 基于关系代数等价变换的查询优化处理

4.1 基本原理和实现方法

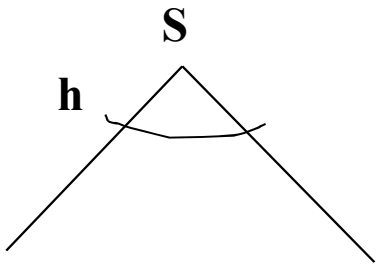
- 实现步骤和方法
  - 转换一：查询问题——〉关系代数表达式
  - 转换二：关系代数表达式——〉查询树
  - 转换三：全局查询树分拆成片段查询树
  - 优化：利用关系代数等价变换规则的优化算法，优化查询树，进而优化查询

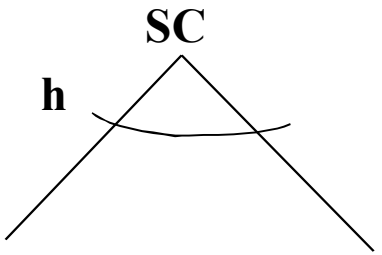
4 基于关系代数等价变换的查询优化处理

4.2 查询优化处理举例

全局关系

S(S#, SNAME, AGE, SEX)和SC(S#, C#, GRADE)被水平分片





S1: SEX='M'　　**S2: SEX='F'**　　**SC1：C#<=20**　　**SC2：C# >20**

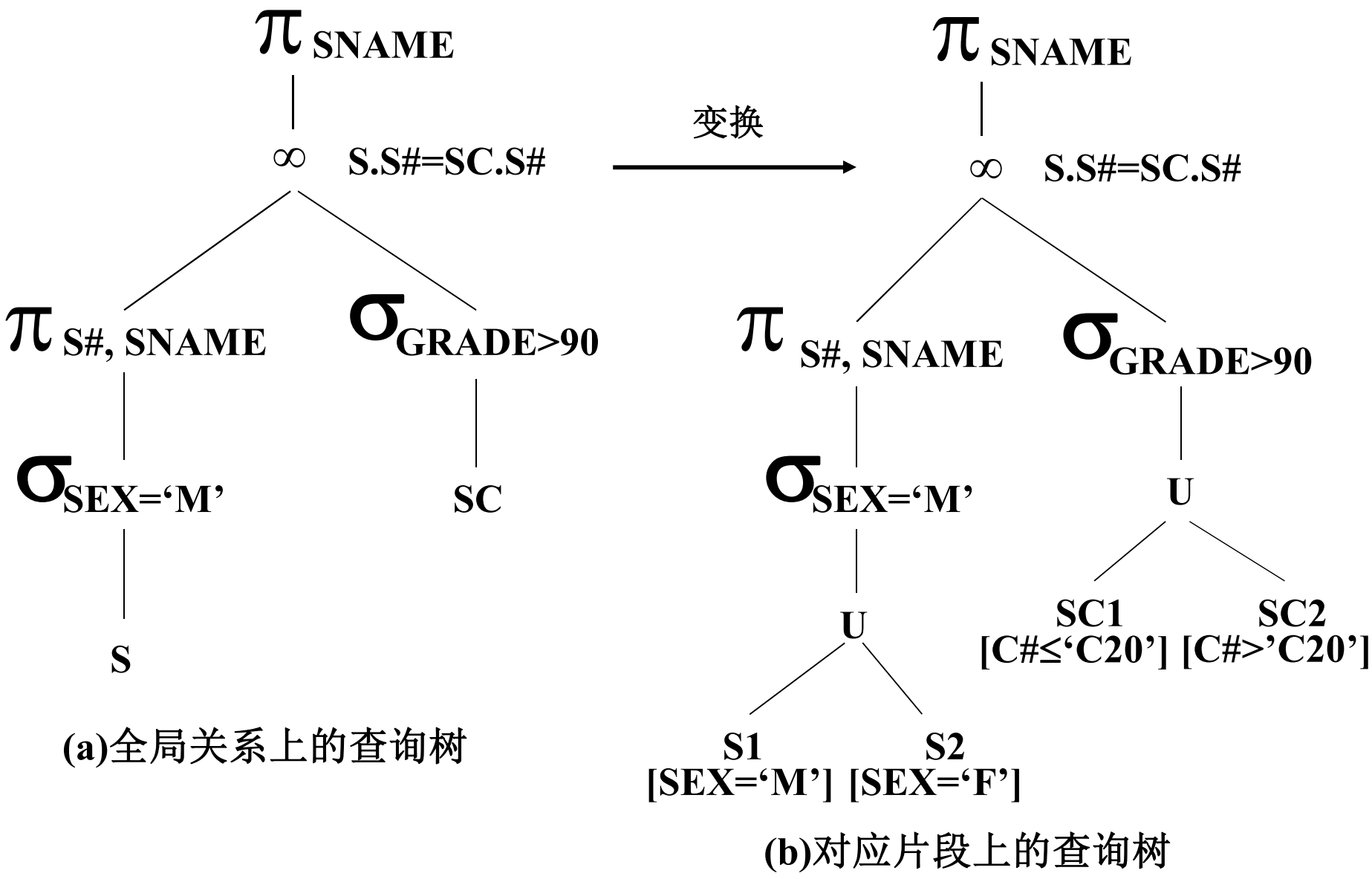
男学生全体　　女学生全体　　课程号<=20　　课程号>20

查询问题：查找至少有一门功课成绩在**90**分以上的男生姓名

$\pi_{SNAME}(\sigma_{SEX='M' \text{ and } GRADE>90}(\sigma_{S.S\ \#=SC.s\#}\ (S\times SC)))$

4 基于关系代数等价变换的查询优化处理

4.2 查询优化处理举例

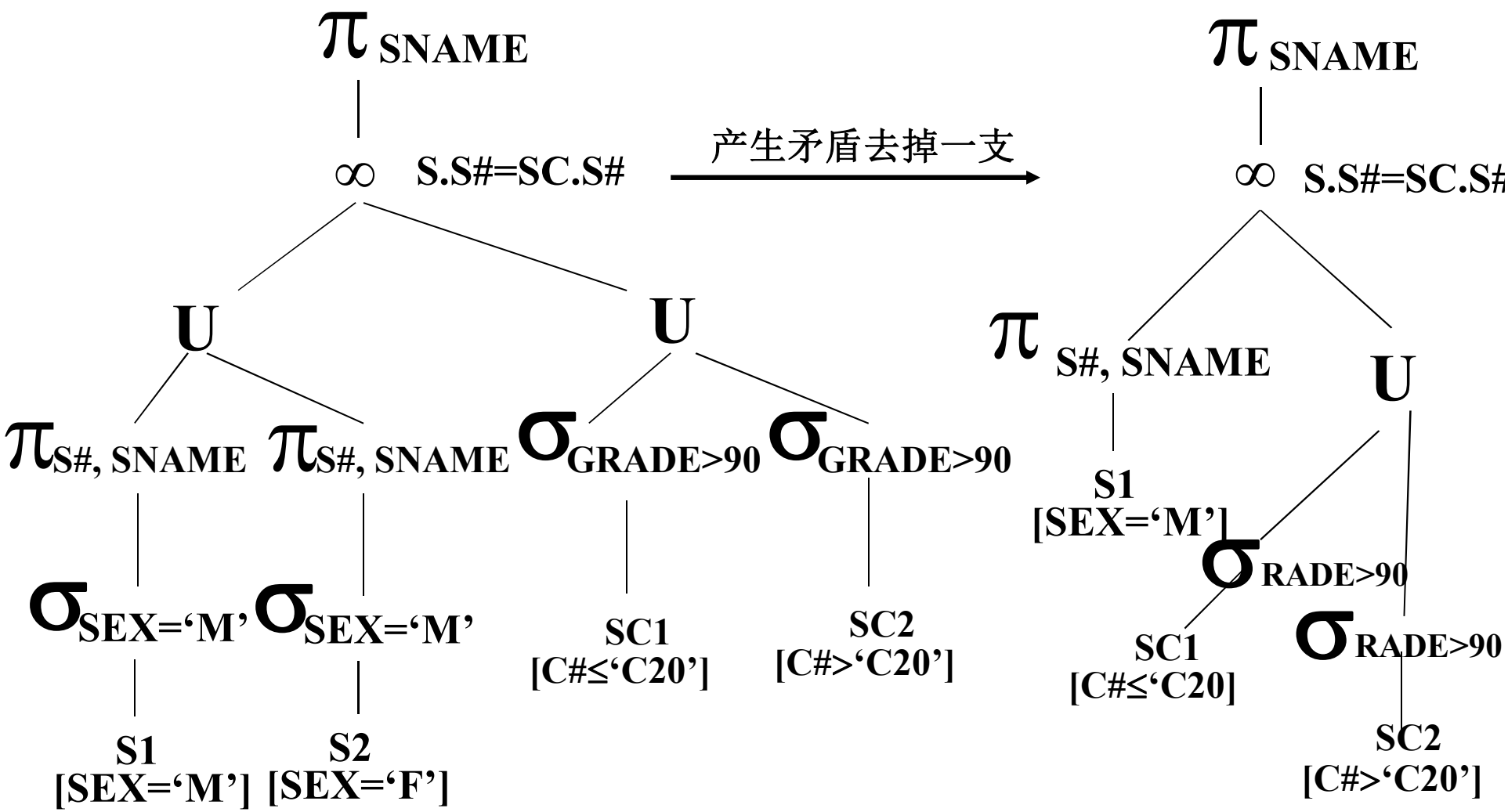


(a)全局关系上的查询树

(b)对应片段上的查询树

4 基于关系代数等价变换的查询优化处理

4.2 查询优化处理举例



（c）把投影和选择下移后的查询树

(d)一个简化的查询树

4 基于关系代数等价变换的查询优化处理

4.2 查询优化处理举例

水平分片的查询优化的基本思想：

1. 尽量把选择条件下移到分片的限定关系处
2. 再把分片的限定关系与选择条件进行比较
3. 去掉它们之间存在矛盾的相应片断
4. 如果最后剩下一个水平片断，则重构全局关系的操作中，就可去掉“并”操作（至少可以减少“并”操作的次数）

4 基于关系代数等价变换的查询优化处理

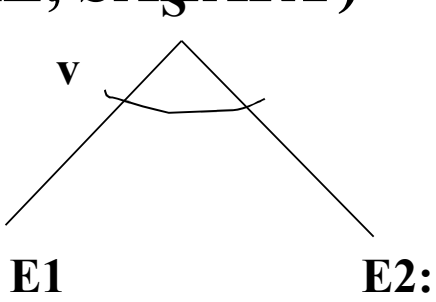
4.2 查询优化处理举例（垂直分片）

全局关系

EMP(EMP#, ENAME, SALARY,DEPT#,DNAME)

垂直分片：**E1 (EMP#,DEPT#,DNAME)**

EMP2(EMP#, ENAME, SALARY)

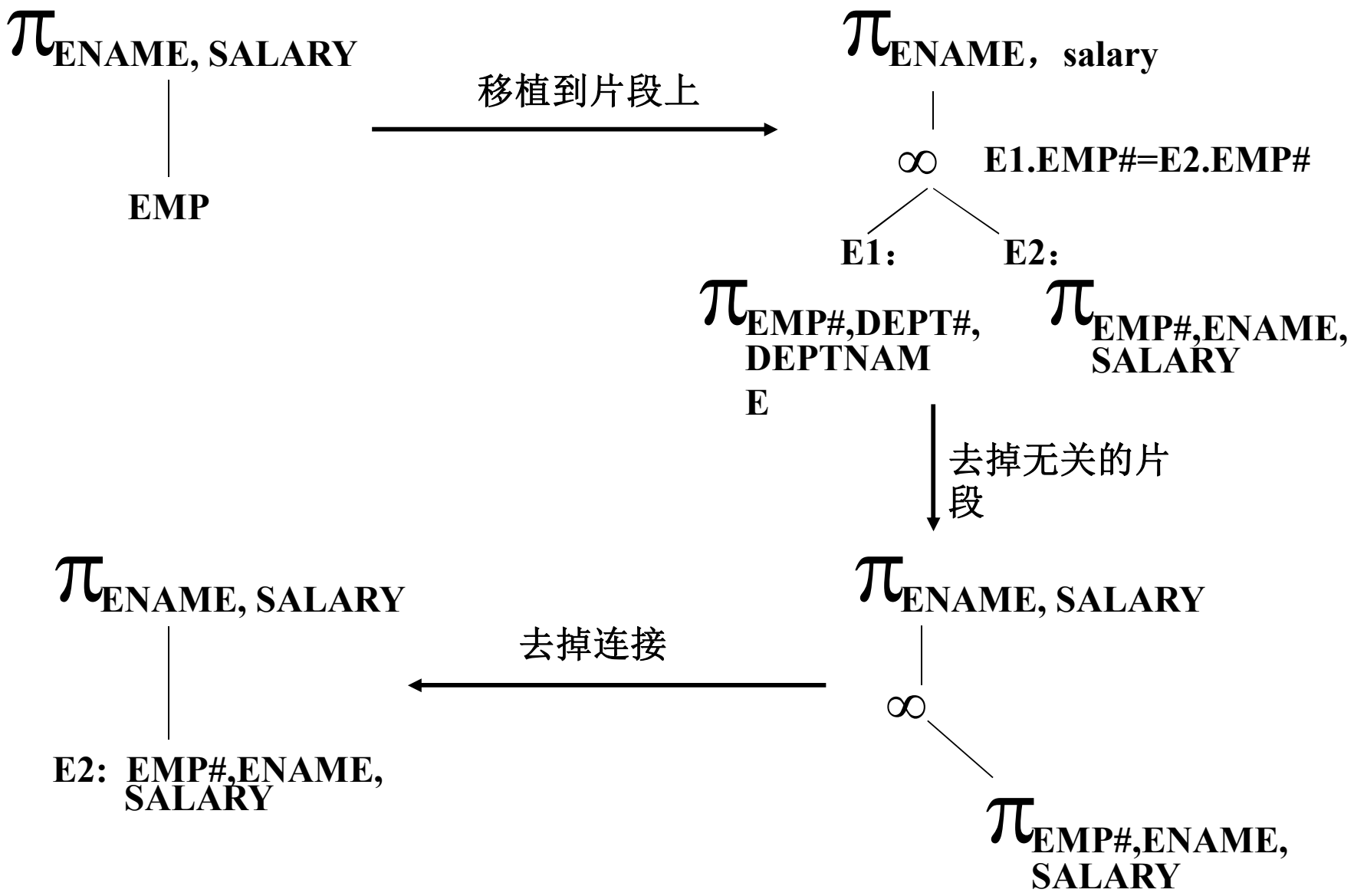


查询问题：雇员的姓名和工资情况

$\pi_{ENAME,SALARY}(EMP)$

4 基于关系代数等价变换的查询优化处理

4.2 查询优化处理举例



4 基于关系代数等价变换的查询优化处理

4.2 查询优化处理举例

垂直分片的查询优化的基本思想：

1. 把垂直分片所用到的属性集，与查询条件中的投影操作所涉及的属性相比较，去掉无关的垂直片断
2. 如果最后只剩下一个垂直片断与查询有关时，去掉重构全局关系的“连接”操作（至少可以减少“连接”操作的次数）

5 基于半连接算法的查询优化处理

5.1 半连接操作

- 假定有两个关系R,S,在属性R.A=S.B上做半连接操作，可表示为：
  - $R \propto_{A=B} S = \pi_R(R \infty_{A=B} S) = R \infty_{A=B} (\pi_B (S))$
  - $S \propto_{A=B} R = \pi_S(S \infty_{A=B} R) = S \infty_{A=B} (\pi_A (R))$
- 用半连接表示连接操作
  - $R \infty_{A=B} S = (R \propto_{A=B} S) \infty_{A=B} S$

  $= (R \infty_{A=B} (\pi_B (S)) \infty_{A=B} S$
  - $R \infty_{A=B} S = (S \propto_{A=B} R) \infty_{A=B} R$

  $= (S \infty_{A=B} (\pi_A (R)) \infty_{A=B} R$

例子1: $R \propto S$

R

A	B
2	a
10	b
25	c
30	d

S

A	C
3	x
10	y
15	z
25	w
32	x

$R \underset{A=A}{\propto} S$

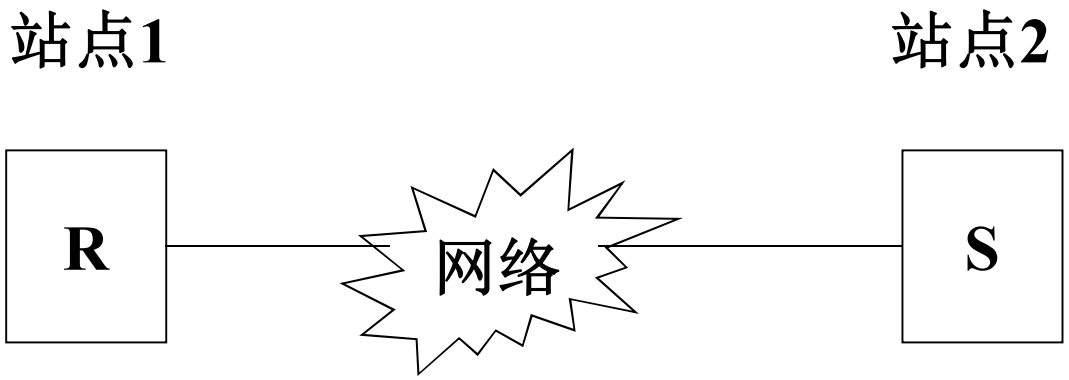
A	B
10	b
25	c

$S \underset{A=A}{\propto} R =$

A	C
10	y
25	w

5 基于半连接算法的查询优化处理

5.2 半连接表示连接的代价估算



(1) $\pi_B(S)$

(3) $R' = R \propto_{A=B} \pi_B(S)$

(2) $\pi_B(S)$

(4) $R' = R \propto_{A=B} \pi_B(S)$

(5) $R' \infty_{A=B} S$

5.2 半连接表示连接的代价估算

- 关系的概貌
  - Card(R)　　片段关系R的元组数目
  - Size(A)　　属性A的大小(即字节数)
  - Size(R)　　片段关系的大小, 属性大小之和
  - Val(A[R])　属性A在R中出现的不同值的个数

5.2 半连接表示连接的代价估算

- 代数操作对关系概貌的影响
  - 选择操作

    $S=\sigma_F(R)$

    $Card(S)=\rho * Card(R)$　　$Size(S)=Size(R)$
  - 并操作

    $T=R \cup S$

    $Card(T) \leq Card(R)+Card(S)$

    $Size(T)=Size(R)=Size(S)$

    $Val(A[T]) \leq Val(A[R])+Val(A[S])$

5.2 半连接表示连接的代价估算

- 代数操作对关系概貌的影响
  - 连接操作

    $T=R\infty S$

    $Card(T) \leqq (Card(R)*Card(S))/Val(A[R])$

    $Size(T) = Size(R)+Size(S)$

    $Val(A[T]) \leq Min(Val(A[R]), Val(A[S]))$　A 是连接属性

  - 半连接

    $T=R\propto S$

    $Card(T) = \rho *Card(R)$

    $Size(T) =$ 第一个操作数Size（R）

    $Val(A[T]) = \rho *Val(A[R])$

    $\rho \approx Val(A[S])/Val(Dom(A))$

5 基于半连接算法的查询优化处理

5.2 半连接表示连接的代价估算

- 代价公式：$T=C_0+C_1*X$
  1. 在站点2上做投影π_B (S)
  2. 把π_B (S)传到站点1上，代价为：
     - C_0+C_1* size (B)* val(B[S])
  3. 在站点1上计算半连接，$R'=R \propto_{A=B} S$
  4. 把R’从站点1传到站点2的代价为：
     - C_0+C_1* size (R’)* card(R’)
  5. 在站点2上执行连接操作：$R' \infty_{A=B} S$
- 采用半连接的总代价
  - $T_{半R}= 2C_0+C_1*$ (size (R’)* card(R’) +size (B)* val(B[S]))
  - $T_{半S}= 2C_0+C_1*$ (size (S’)* card(S’) +size (A)* val(A[R]))
- 比较$T_{半R}$ 与$T_{半S}$，取最优者

5.3 半连接算法优化原理和步骤

- 基本原理
  1. 通常有两次传输
  2. 但是传输的数据量和传输整个关系相比，要远远少
  3. 一般有：$T_{半} << T_{全}$
  4. 半连接的得益：当card（R）>>card（R’），可减少站点间的数据传输量
  5. 半连接的损失：传输$\pi_B (S) = C_0 + C_1 * size(B) * val(B[S])$
  6. 基本原理是在传到另一个站点做连接前，消除与连接无关的数据，减少做连接操作的数据量，从而减小传输代价
- 采用半连接优化算法的步骤
  - 计算每种半连接方案的代价，并从中选择一种最佳方案
  - 选择传输代价最小的站点，计算采用全连接的方案的代价
  - 比较两种方案，确定最优方案

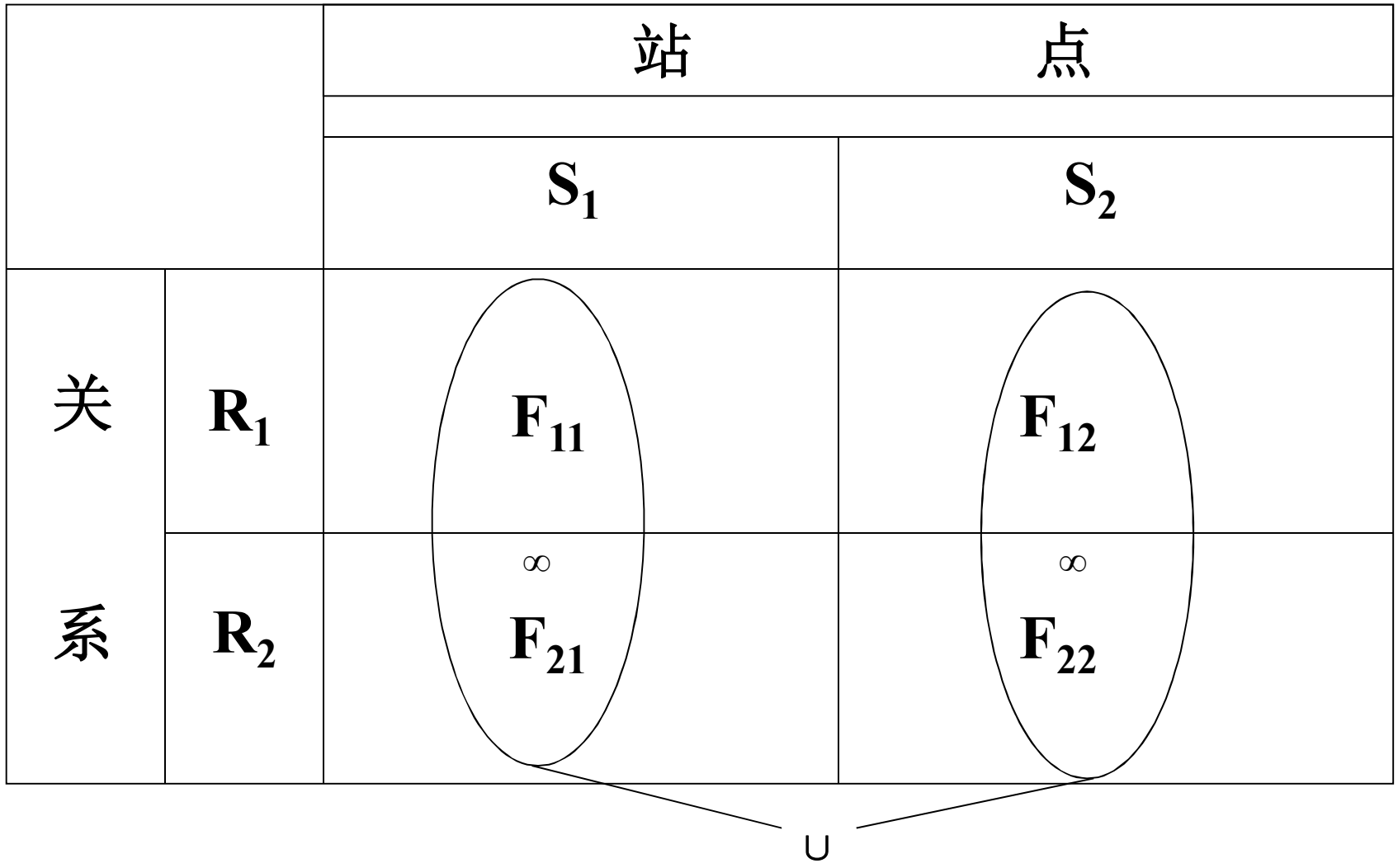
6 基于直接连接算法的查询优化处理

6.1 概述

- 半连接算法和直接连接算法区别
  - 取决于数据传输和局部处理的相对费用
  - 如果传输费用是主要的，采用半连接
  - 如果本地费用是主要的，采用直接连接
- 几种基于直接连接的优化算法 （考虑关系分段）
  - 利用站点依赖信息的算法
  - 分片与复制算法
  - Hash划分算法

6 基于直接连接算法的查询优化处理

6.2 利用站点依赖信息的算法



6.2 利用站点依赖信息的算法

- 站点依赖

  设关系R_i分片F_{i1}和F_{i2}, R_j分片F_{j1}和F_{j2}

  关系R_i和R_j在属性A上满足条件

  $$F_{is} \propto_A F_{jt} = \varnothing \text{, 其中} s \neq t,$$

  则称R_i和R_j在属性A上站点依赖

  也就是说:

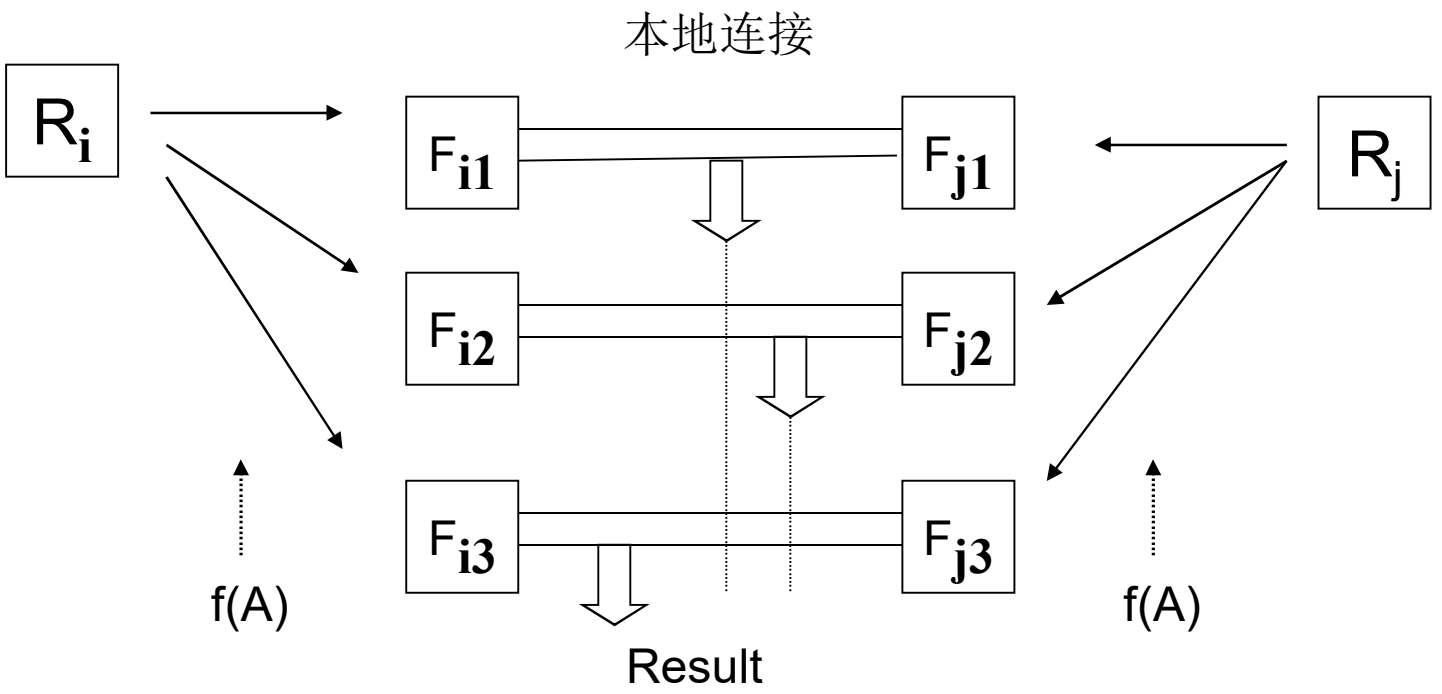
  $R_i \propto_A R_j = U(F_{is} \propto_A F_{js})$, 对于包含着两个关系的片段的每个站点s都成立

  此时关系的连接操作无站点间数据传输

6 基于直接连接算法的查询优化处理

6.2 利用站点依赖信息的算法

$R_i \underset{A}{\infty} R_j$



6.2 利用站点依赖信息的算法

- 推论
  - 1 若R_i和R_j在属性A上站点依赖，则R_i和R_j在任何包含A的属性集B上也站点依赖。
  - 2 若R_i和R_j在属性A上站点依赖，另一属性（或属性组）B函数决定A，且$A \neq \varnothing$，则R_i和R_j在B上也站点依赖。
  - 3 若R_i和R_j在属性A上站点依赖，且若R_j和R_k在属性B上站点依赖，则（$R_i \infty_A R_j \infty_B R_k$）=U（$F_{is} \infty_A F_{js} \infty_B F_{ks}$）；查询$R_i \infty_A R_j \infty_B R_k$的连接操作能够以无数据传输的方式处理。

6 基于直接连接算法的查询优化处理

6.2 利用站点依赖信息的算法

- 算法描述
  - Placement_Dependency (Q, P, S)，其中：
    - R={R1,R2,R3,…,Rn}是查询Q引用的一组关系
    - P是站点依赖信息
    - S是一个连接操作可以无数据传输的执行的最大关系集合
    - 开始时S是空集。算法结束时，**若S=R,则Q可以无数据传输执行**
  - 算法步骤
    - 初始化S= ∅， R={R1,R2,R3,…,Rn}
    - 若能找到一对关系R_i和R_j在属性A上站点依赖，且$R_i \propto_C R_j$包含在Q中，其中C包含A,那么把R_i和R_j放到S中，否则算法终止，返回空集S。
    - 只要存在R中而不在S中的关系R_k满足下面的特性，就把其放入S中：有S中的关系比如R_j，与R_k在属性B上有站点依赖关系，且$R_j \propto_B R_k$在Q中或者可以由Q导出，根据推论3，则R_k可被包含在S中。

6 基于直接连接算法的查询优化处理

6.3 分片和复制算法

		站点	
		S_1	S_2
关系	R_1	F_{11}	F_{12}
	R_2	R_2	R_2

6.3 分片和复制算法

- 查询引用的某个关系的所有片段分布在这些站点上,其余被引用的关系复制到每一个选定的站点

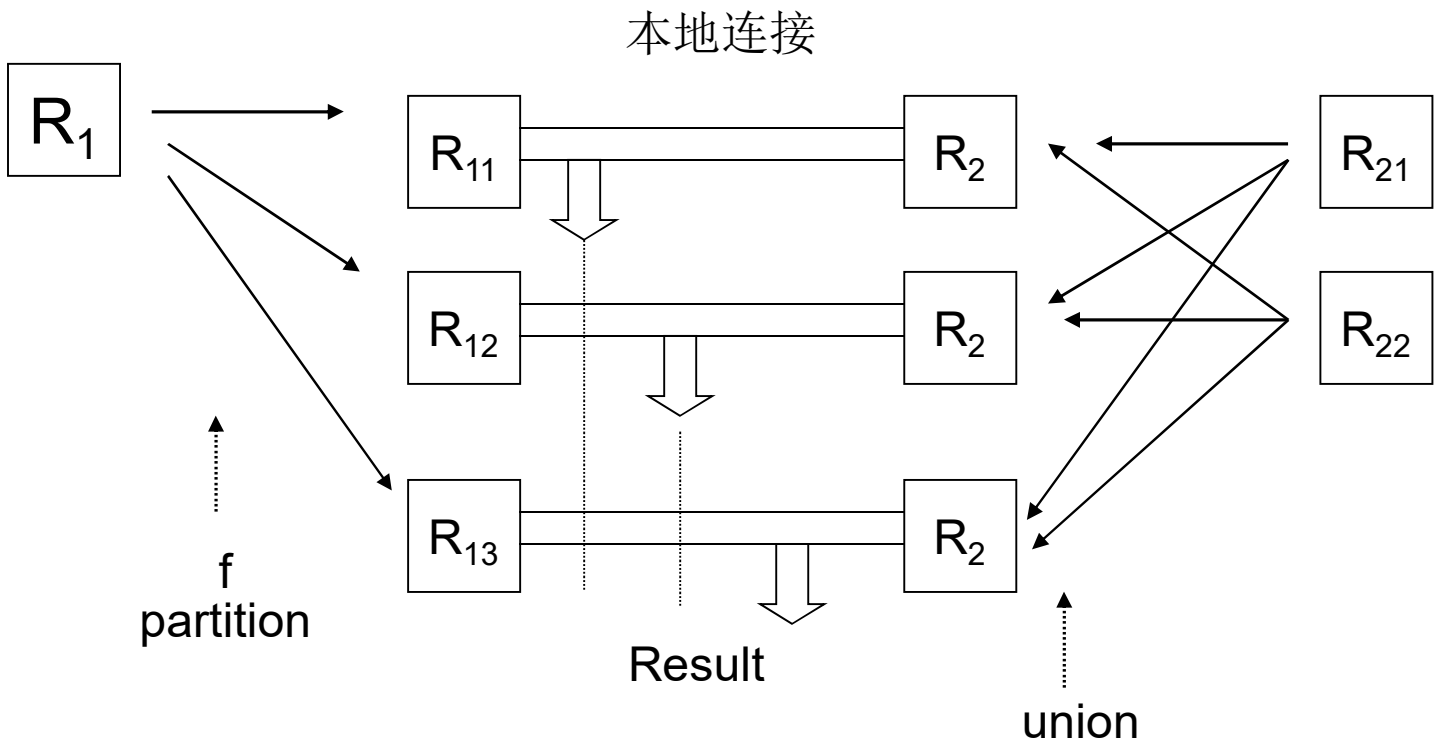
$$R_1 \propto R_2 = U_i (F_{1i} \propto R_2)$$

- 算法可应用到涉及两个或两个以上的关系的查询
  - 其中一个关系保持分片状态
  - 其他关系可先连接起来，再被复制到各个站点
  - 在各个站点上，其他关系副本与相应的第一个关系的片断连接
  - **要求确定那个分片关系保留**

6 基于直接连接算法的查询优化处理

6.3 分片和复制算法

$R_1 \quad \infty \quad R_2$



6 基于直接连接算法的查询优化处理

6.3 分片和复制算法

Fragmentation_and_replicate(Q, R, S)

For 每个保持分片状态的关系R_i

For 每个包含关系R_i的一个片段的站点S_j

计算在站点S_j执行子查询的完成时间

$FT(Q, S_j, R_i)$

计算关系Ri保持分片状态下的响应时间

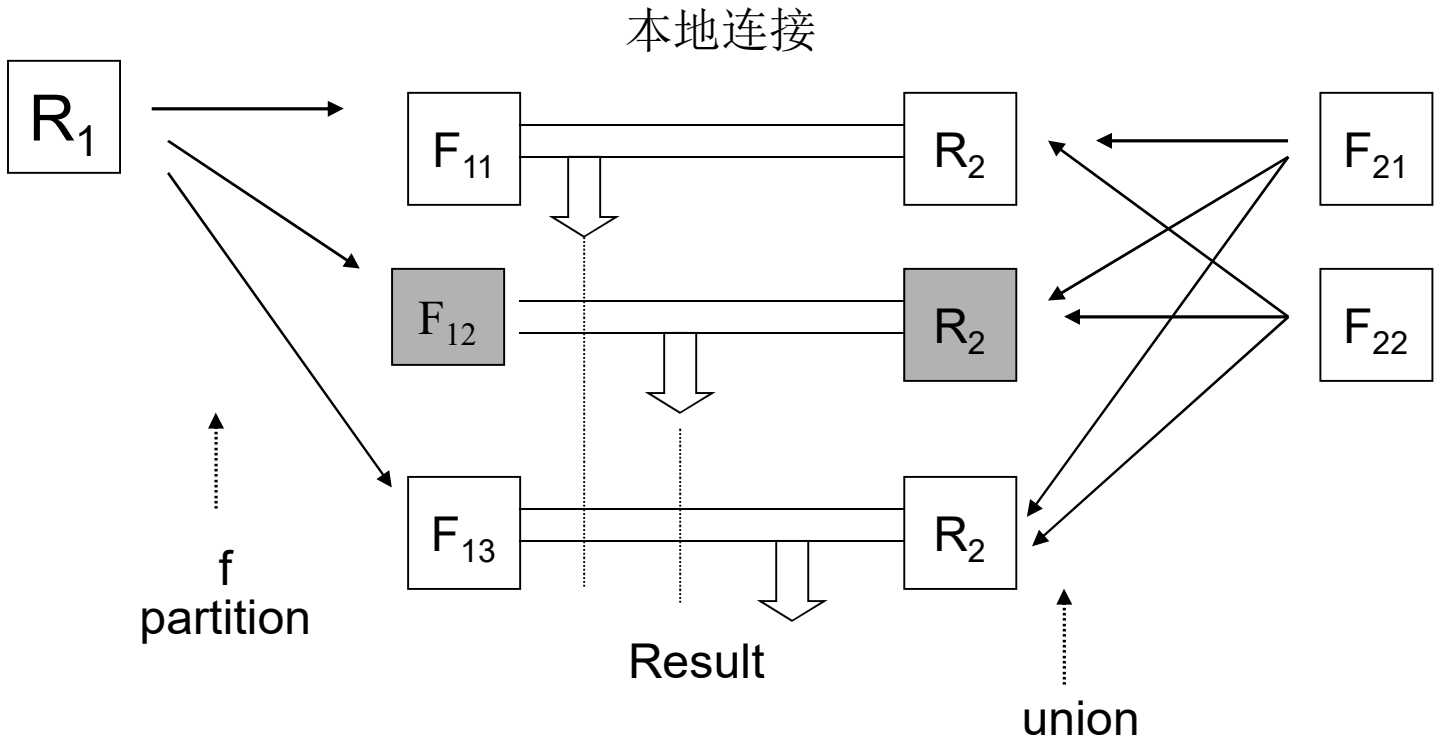
$T_i = \max_j (FT(Q, S_j, R_i))$

选择$R_k = \min_i (T_i)$为保持分片状态的关系

6 基于直接连接算法的查询优化处理

6.3 分片和复制算法

$R_1 \infty R_2$



6 基于直接连接算法的查询优化处理

6.3 分片和复制算法

举例

已知R_1分段F_{11}和F_{12}的大小为: $|F_{11}|=|F_{12}|=50$

R_2分段F_{21}和F_{22}的大小为: $|F_{21}|= 100$　$|F_{22}|=200$

设　数据通讯$C_0=0, C_1=1$ （参照半连接优化定义）

本地连接$Cost=J(x_1, x_2)=5*(x_1+x_2)$

并操作$Cost = U(x_1, x_2) = 2*(x_1+x_2)$

令R_1保持分片状态, 则: 站点S_1的完成时间

$FT(Q, S_1, R_1) = 200+2*(100+200)+5*(50+300)=2550$

同理: $FT(Q, S_2, R_1) = 100+2*(100+200)+5*(50+300)=2450$

因此, 查询响应时间在R_1保持分片状态为 2550.

6.3 分片和复制算法

令R_2保持分片状态, 则: 站点S_1的完成时间

$$FT(Q, S_1, R_2) = 50+2*(50+50)+5*(100+100)=1250$$

同理:

$$FT(Q, S_2, R_2) = 50+2*(50+50)+5*(200+100)=1750$$

因此, 查询响应时间在R_2保持分片状态为 1750.

因为:

R_1保持分片状态的响应时间>R_2保持分片状态的响应时间

所以: 选择R_2保持分片计算查询

6.4 Hash划分算法

- 利用Hash函数对分片关系上的连接属性作站点依赖计算, 再据此分片, 以获取站点依赖的连接算法
- 例如, 运用Hash函数

$$h(a)=\begin{cases}1 & \text{若a 是奇数}\\ 0 & \text{若a 是偶数}\end{cases}$$

对R中每个元组, h(a)为1送入站点S_1, h(a)为0送入站点S_2. 于是片段关系R被划分为R^o和R^e

- $R_1 \infty R_2 = (R_1^o \infty R_2^o) \cup (R_1^e \infty R_2^e)$

6 基于直接连接算法的查询优化处理

6.4 Hash划分算法

- 利用Hash函数对分片关系上的连接属性作站点依赖计算，再据此分片，以获取站点依赖的JN算法
- 例如，运用Hash函数

$$h(a) = \begin{cases} 1 & \text{若a 是奇数} \\ 0 & \text{若a 是偶数} \end{cases}$$

- 片断F_{11}按属性A的值为奇和偶数划分成F_{11}^{o}和F_{11}^{e}，片断F_{12}划分成F_{12}^{o}和F_{12}^{e}
- 站点S_2上$F_{12}' = F_{11}^{e} \cup F_{12}^{e}$，站点$S_1$上$F_{11}' = F_{11}^{o} \cup F_{12}^{o}$
- 显然$\pi_A(F_{11}')$和$\pi_A(F_{12}')$没有公共值，前面是奇数值后面是偶数值
- $F_{12}' \propto F_{11}'$是空集，这说明R_1和R_2在新组成的片断下在属性A上站点依赖。
- $R_1 \propto R_2 = (F_{11}' \propto F_{21}') \cup (F_{12}' \propto F_{22}')$

6.4 Hash划分算法

- 考察三个关系R_1，R_2和R_3 ,它们在两个站点上，有两种情况：
  - 在同一属性A上连接，$R_1 \infty_A R_2 \infty_A R_3$
    - 在三个关系的片断上应用Hash函数
    - 使用新组建的片断，三个关系在属性A上将满足站点依赖
    - 经这种划分和数据传送之后，两个站点上的片断在属性A上的连接就可以并行进行，合并执行结果给出答案
  - 在不同属性上连接， $R_1 \infty_A R_2 \infty_B R_3$

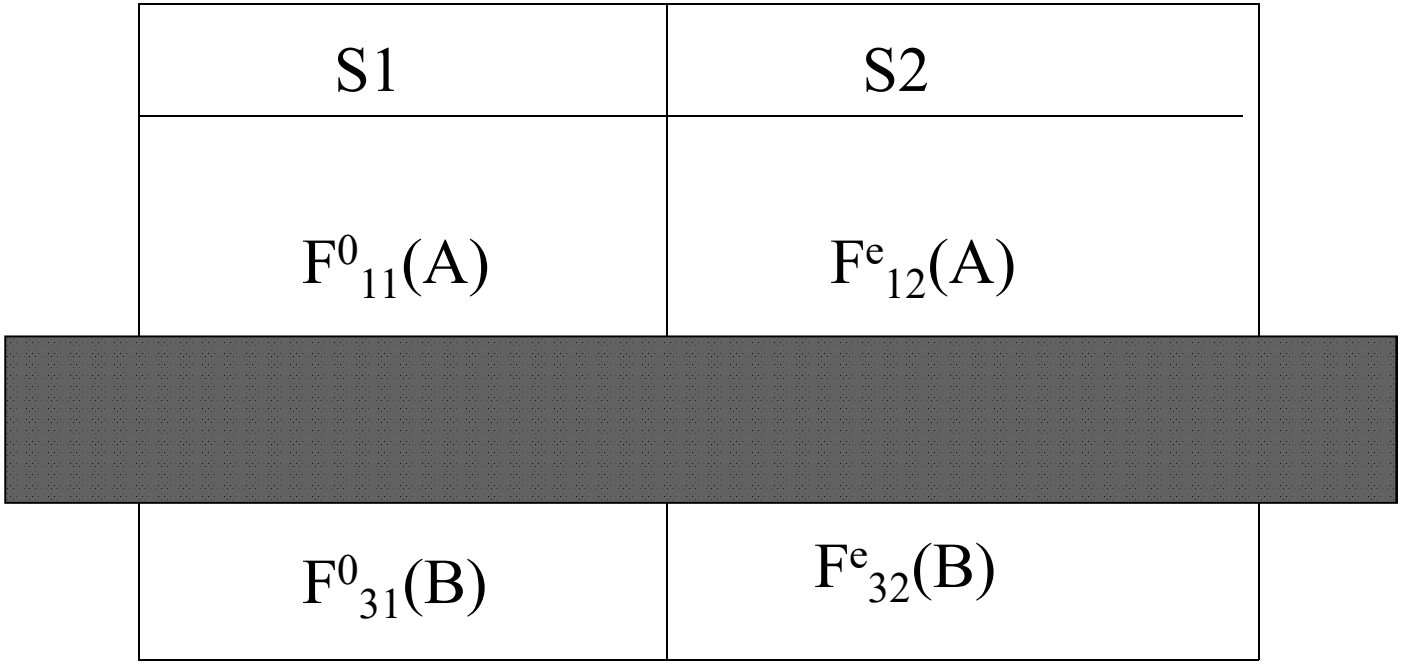
6.5 Hash划分算法

- 两种情况：
  - 在同一属性A上连接，$R_1 \infty_A R_2 \infty_A R_3$
  - 在不同属性上连接，$R_1 \infty_A R_2 \infty_B R_3$
    - 在属性A上应用同样的Hash函数，在属性B上也应用同样的Hash函数，可能得不到希望的站点依赖
    - 因R1中属性A的值是奇数的发往S1，R3中属性B是奇数的元组发往S1.但R2中某些元组可能在A上有奇数值，而在B上有偶数值
    - 解决方法是允许这些元组在两个站点上都存在。

6.5 Hash划分算法

- $R1 \propto_A R2 \propto_B R3$

  若R1与R2在A上有相同的Hash函数, R2与R3在属性B上有相同的Hash函数



6 基于直接连接算法的查询优化处理

6.5 算法比较(两个关系)

- 站点依赖算法
  - 无数据传递
  - 可利用索引做本地连接
  - 每个站点连接数据总量是R，两个片段
- 分片和复制算法
  - 数据传输总量是R
  - 每个站点的连接数据量是(3/2)R,一个全关系和一个片断
- Hash划分算法
  - 数据传送量是R
  - 每个站点的连接数据量同站点依赖

		站点	
		S_1	S_2
关系	R_1	F_{11}	F_{12}
	R_2	F_{21}	F_{22}

假定每个片段的大小是R大小的一半R/2

练习1

有关系R，S，T，如图所示

1. 计算连接R ∞ S ∞ T
2. 计算半连接R∝S，S∝R，S∝T，T∝S，R∝T，T∝R

R

A	B	C
2	3	5
5	3	6
1	6	8
3	4	6
5	3	5
2	6	8

S

B	C	D
3	5	6
3	5	9
6	8	3
5	9	6
4	1	6
5	8	4

T

D	E	F
6	6	9
8	7	8
8	5	6
3	8	9

练习2

在如下R, S的概貌上计算R $\infty_{A=B}$ S

Size(R)=50, Card(R)=100, Val(A[R])=50, Size(A)=3

Size(S)=5, Card(S)=50, Val(B[S])=50, Size(B)=3

R $\propto_{A=B}$ S 的选择度 $\rho = 0.2$

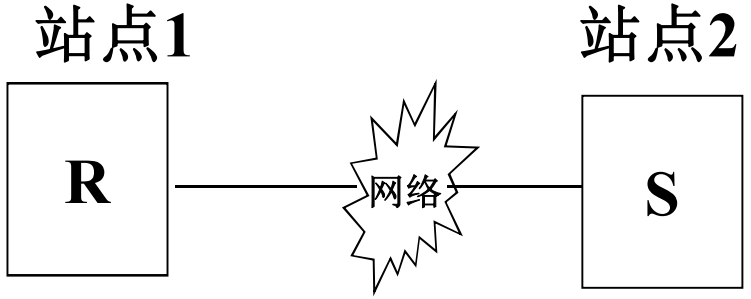
S $\propto_{A=B}$ R 的选择度 $\rho = 0.8$

$C_0=0, C_1=1$

问:

1. 使用 $\propto$ 简化程序在R的站点执行∞
2. 使用 $\propto$ 简化程序在S的站点执行∞
3. 使用直接连接在R站点执行∞
4. 使用直接连接在S站点执行∞

那种方案较优?



总　结

- 分布式查询优化概述（目标、准则和代价估算）
- 基础知识
  - ✓关系代数回顾
  - ✓等价变换规则
- 分布式查询分类和层次结构
- 基于关系代数等价变换的查询优化
- 基于半连接算法的查询优化处理
- 基于直接连接算法的查询优化处理