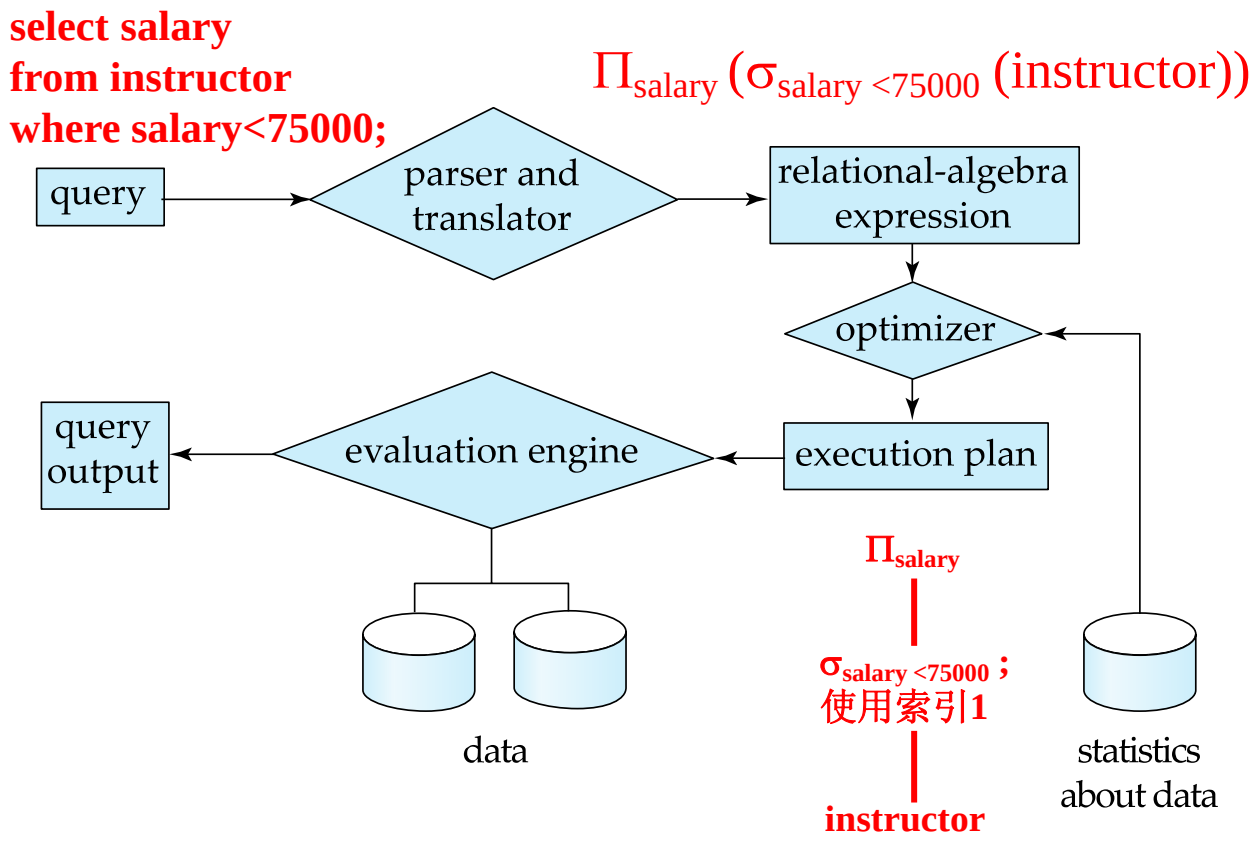
第七章 关系查询处理与查询优化

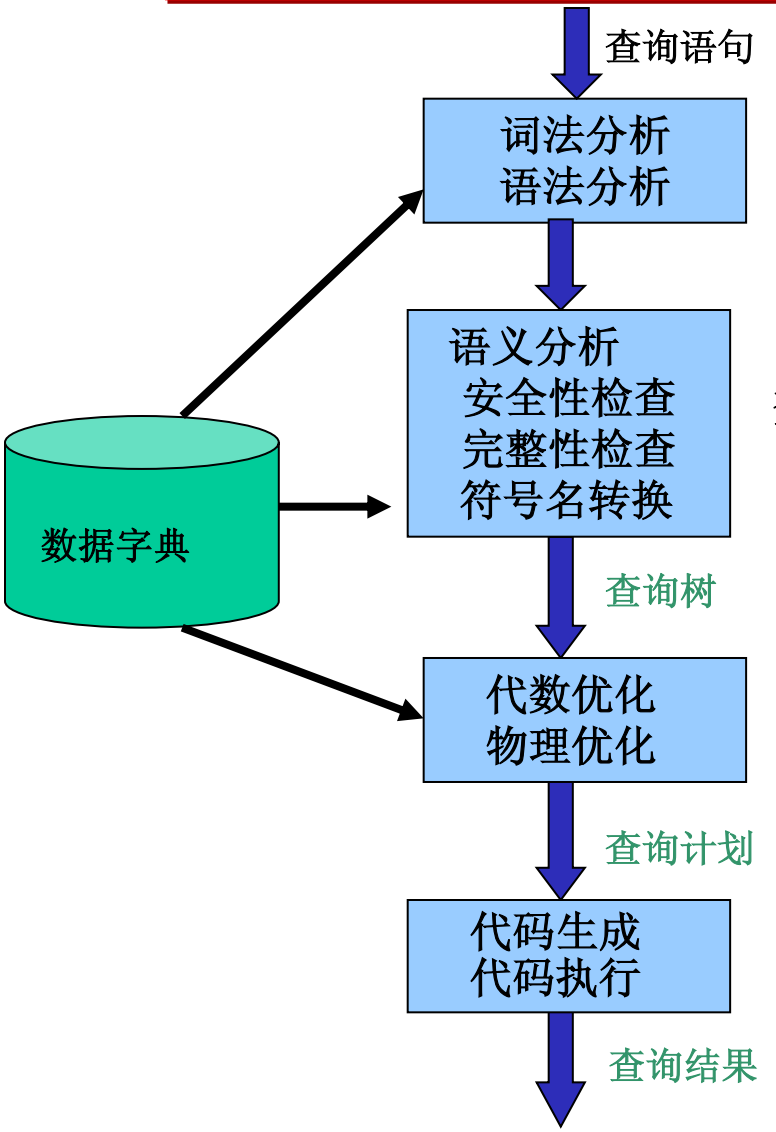
- 查询处理概述
- 查询操作的实现
- 查询优化

关系查询处理步骤

- 关系查询处理的四个阶段：
  - 查询分析
  - 查询检查
  - 查询优化
  - 查询执行



关系查询处理步骤



查询分析

- 词法分析
- SQL语法检查和语法分析

查询检查

- 语义检查
- 存取权限检查
- SQL语句转换为关系代数表达式（查询树/语法分析树）

查询优化

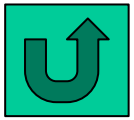
- 选择一个高效执行的查询处理策略
- 生成查询计划

查询执行

- 生成查询计划的代码
- 代码执行

查询代价的度量

- 查询时间开销与多个因素有关
  - 磁盘访问，CPU, 网络通信等
- 对于驻留在磁盘上的大型数据库，从磁盘访问数据的I/O代价通常是最重要的代价
- 假设存取一个块就需要一次磁盘访问，使用访问磁盘的块数作为估计代价的因素
  - t_T –传输一个块的平均时间
  - t_S – 一次磁盘寻道时间加上旋转延迟
  - t_T 和 t_S 与磁盘类型有关，例如对于4KB的块：
    - 高端磁盘：t_S =4 毫秒 ， t_T =0.1毫秒
    - SSD: t_S = 90 微秒 ， t_T =10微秒



查询操作的实现

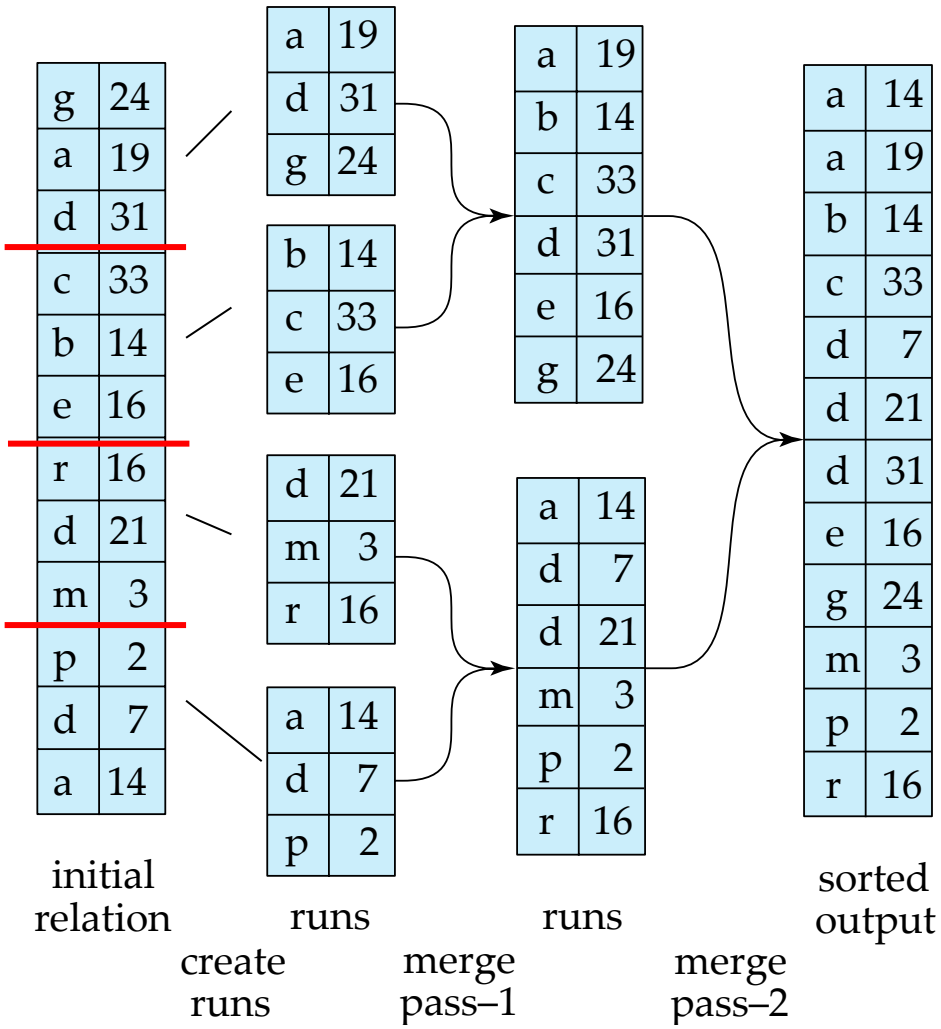
- 选择运算实现算法
- 连接运算实现算法
- 排序
- 其他运算

实现查询操作的算法—选择

- 选择(选取)操作的实现算法
  - 全表扫描法
    - 按照物理顺序读表的M块到内存，检查内存的每个元组t，如果满足条件则输出t，直到表所有块都经过上述检查
  - 索引扫描法
    - 如果在选择条件的属性上有索引，先通过索引找到目标索引项，再通过索引项找到元组

实现查询操作的算法-排序

- 内存中完全容纳的关系，可用快速排序法(quicksort)等算法；内存中无法容纳的关系，采用外排序-归并算法
- 外排序-归并
  - 第一阶段，建立多个排好序的归并段（run），每个段仅包含关系的部分记录；
  - 第二阶段对归并段进行归并



外排序-归并法示例

实现查询操作的算法—连接

- 连接操作的实现算法示例
  - 嵌套—循环法
    - 两个连接的表，第一个表为外循环，另一个为内循环
  - 索引连接法
    - 第二个表按照连接属性建索引，取第一个表元组的连接属性与第二个表元组的连接属性比较
  - 排序—合并法
    - 两个表都按照连接属性排序，取第一个表元组的连接属性与第二个表元组的比较
  - Hash join法
    - 连接属性作为hash码，用同一个hash函数把两个连接表的元组散列到同一个hash文件

实现查询操作的算法—连接

- 嵌套-循环法
  - 两个连接的表，一个表为外循环，另一个为内循环
  - 连接运算 $r \bowtie_{\theta} s$的实现算法，r是外循环表，s是内循环表：

for each 元组 t_r in r do begin
  for each 元组 t_s in s do begin
    测试元组对(t_r,t_s)是否满足连接条件θ，
    如果满足，则把 $t_r \cdot t_s$ 加入到结果中.
  end
end

查询示例：

```
SELECT   S#,SN, C#, G
FROM   S,  SC
WHERE  S.S#=SC.S#
```

  - 不需要索引，并适用于任何连接条件
  - 需要检查两个关系中的每一对元组，代价高

实现查询操作的算法—连接

- 嵌套-循环法
  - 如果连接操作使用的缓冲区块数为k，分配（k-1）块给外表r，1块给内表s，则存取块数为：

    $b_r + b_r/(k-1) * b_s$ ，

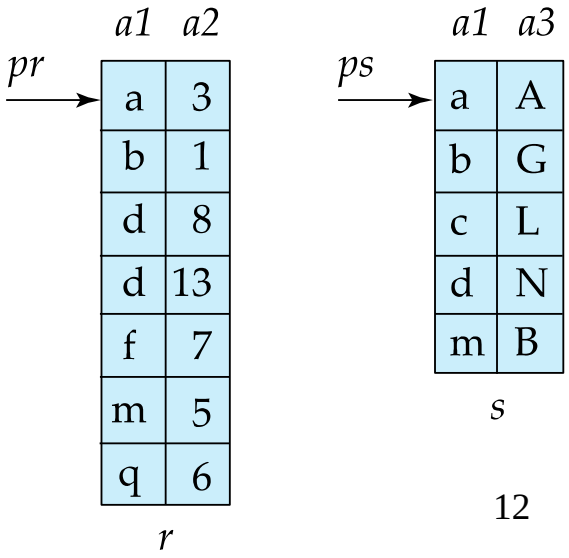
    其中b_r为表r的块数，b_s为表s的块数
  - 应选择较小表作为外表

实现查询操作的算法—连接

- 索引连接法（索引嵌套-循环连接法）
  - 如果内层关系s的连接属性上有索引，则对于外层关系r中的每一个元组 t_r，可以用索引查找s中与 t_r 满足连接条件的元组
  - 如果r和s在连接属性上都有索引，则以元组较少的关系作为外层关系，代价最小

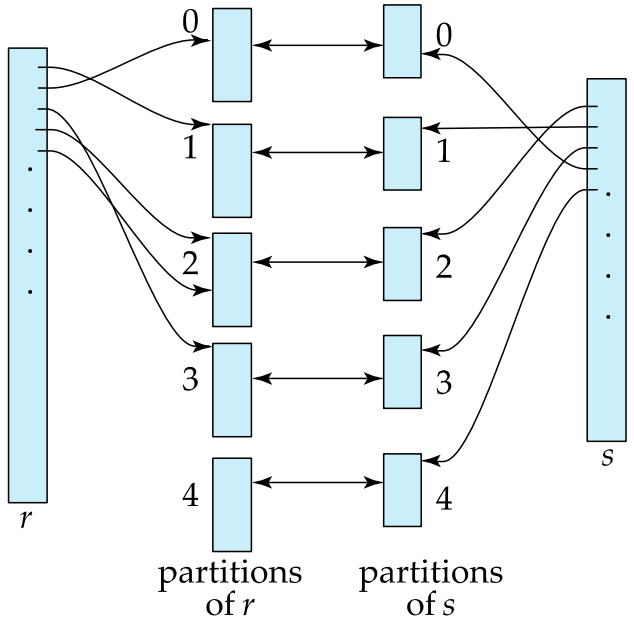
实现查询操作的算法—连接

- 排序-合并法（合并-连接法）
  - 将两个关系在连接属性上排序
  - 为每个关系分配一个指针，分别为p_r和p_s，对于p_r指向r的一个连接属性值，移动p_s找到s中与该值相等的元组，与p_r指向的元组做连接，如此移动 两个指针分别遍历r 和s
  - 两个关系的每个块都只需要读一次，访问块数：
    - 访问块数= $b_r + b_s$
  - 只能用于等值连接或自然连接



实现查询操作的算法—连接

- **Hash join法（哈希连接法）**
  - 哈希函数h用于划分两个关系。h将连接属性值映射到{0, 1, ..., n}集合上。将r的元组划分为$r_0, r_1, \ldots, r_n$ 个部分，如果 $h(t_r[JoinAttrs])=i$, 则元组t_r 将被放入r_i ，同理将s也划分成$s_0, s_1 \ldots, s_n$ 个部分。
  - 对于某个连接属性值，若被哈希为i，则s中相应的元组必定在s_i中，而r中的元组必定在r_i 中，因此只需将 s_i和r_i中的元组相比较
  - 适用于等值连接或自然连接



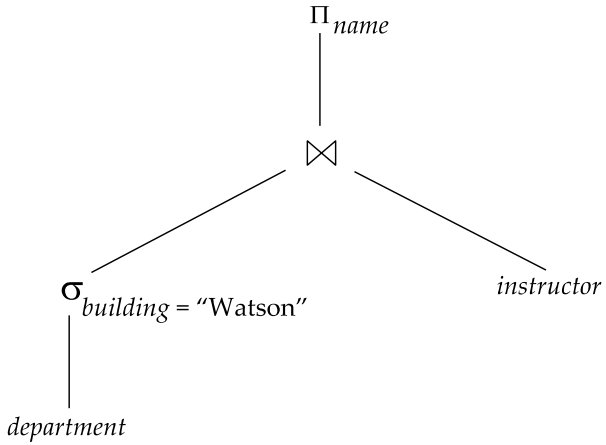
其他运算的实现算法

- 去重：使重复数据相邻，保留一个数据删除其它
  - 排序方法
  - 哈希方法
- 投影：在每个元组上执行投影，之后再去重
- 集合运算—并，差，交
  - 类似排序-合并连接，排序然后对每个已排序的关系扫描一次
  - 类似hash-join将两个关系分区，在两个关系对应区中执行运算
- 库函数
  - 基于分组属性进行排序或散列以聚集同组的元组，再执行库函数

表达式的执行

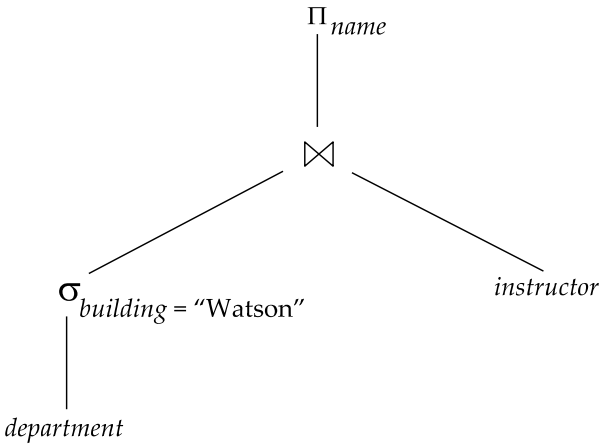
- 可选方法包括物化(materialized)方法和流水线(pipeline)方法
- 物化方法
  - 按次序每次只执行一个运算，运算结果被物化到一个临时关系中，这些临时关系一般需要写到磁盘上
  - 示例，对于查询：$\Pi_{name}(\sigma_{building="\text{Watson}"}(department) \infty instructor)$
    - 先计算 $\sigma_{building="\text{Watson}"}(department)$

      并将结果表存为临时表，再执行连接运算 生成临时表，再执行投影运算。
  - 方法适用性广泛，但临时表的写和读代价大



表达式的执行

- 流水线方法
  - 同时执行多个运算，将结果传递给下一个运算
  - 示例：不存储选择操作的结果，而将结果元组直接传递给连接操作，连接操作将结果传递给投影，不存储临时关系
  - 不需要在磁盘上存储临时关系，代价小，但对有些运算不适用，如排序等



查询优化

- 查询优化的必要性
- 查询优化的目标
- 查询优化方法
  - 查询的代数优化
  - 查询的物理优化
- 查询优化的一般步骤

查询优化的必要性

- 执行语句：

  SELECT SN

  FROM S, SC

  WHERE S.S# = SC.S# AND C#='02';

  – S表中有1000个元组，SC有10000个元组，'02'课程的选课记录为50个；

  – 内存一个块装10个S元组，或100个SC元组；内存放5块S元组，1块SC元组，每块的读取时间为0.05 s。

- 完成方式：

$Q1 = \prod_{SN}(\sigma_{S.S\#=SC.S\# \wedge C\#='02'}(S \times SC))$，

笛卡尔积读写数据块数+存临时关系+取临时关系 $=2100+10^6+10^6 \approx 10^5$ s

$Q2 = \prod_{SN}(\sigma_{C\#='02'}(S \bowtie SC))$

连接读写数据块数+存临时关系+取临时关系 $=2100+10^3+10^3 \approx 205$ s

$Q3 = \prod_{SN}(S \bowtie \sigma_{C\#='02'}(SC))$

读取S,SC数据块 $=100+100 \approx 10$ s

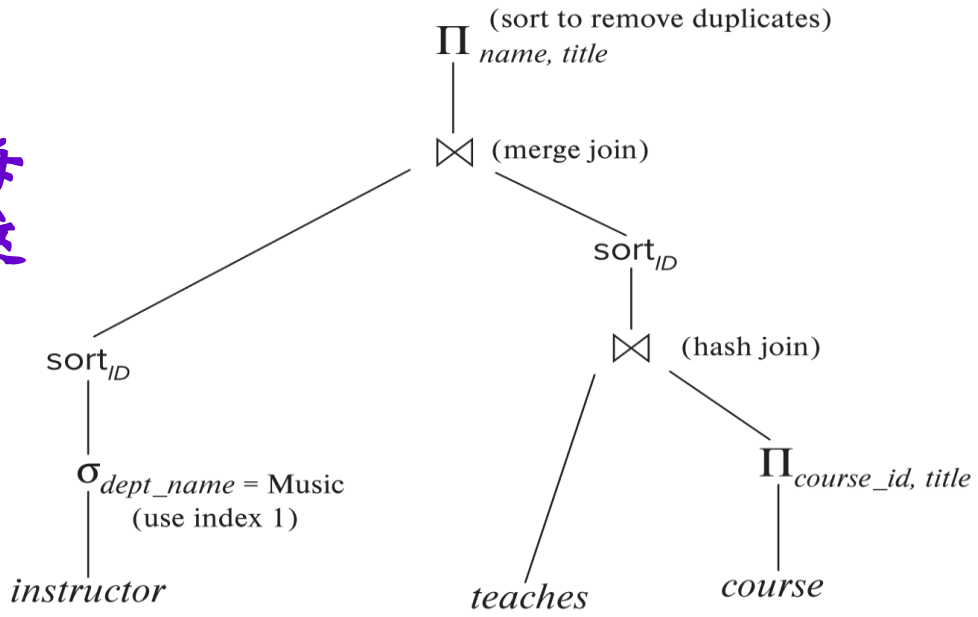
查询优化目标

- 查询的执行开销（集中式数据库）
  - 总代价=I/O代价 + CPU代价 + 内存代价
  - I/O代价是最主要的
- 查询优化目标
  - 选择一个高效执行的查询处理策略，使得查询代价最小，即访问磁盘的块数最少

查询优化

- 按照优化的层次分为代数优化和物理优化
  - 代数优化：关系代数表达式的优化，即按照一定的规则，改变代数表达式中操作的次序和组合
  - 物理优化：存取路径和底层操作算法的选择。包括基于规则或基于代价等

- 查询优化的结果
  - 查询计划：定义了每个操作的算法以及这些操作执行的顺序



代数优化

- 通过对关系代数表达式的等价变换来提高查询效率
- 关系代数表达式的等价
  - 指用相同的关系代替两个表达式中相应的关系所得到的结果是相同的

代数优化

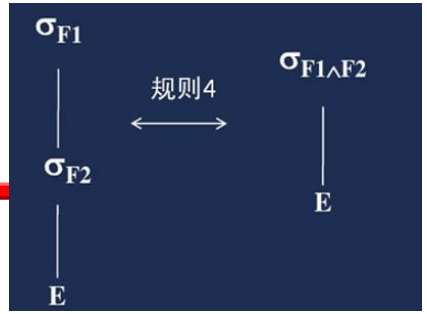
- 关系代数表达式等价变换规则
  - 1.连接、笛卡尔积交换律

$$E_1 \times E_2 \equiv E_2 \times E_1$$
$$E_1 \bowtie E_2 \equiv E_2 \bowtie E_1$$
$$E_1 \underset{F}{\bowtie} E_2 \equiv E_2 \underset{F}{\bowtie} E_i$$

  - 2. 连接、笛卡尔积的结合律

$$(E_1 \times E_2) \times E_3 \equiv E_1 \times (E_2 \times E_3)$$
$$(E_1 \bowtie E_2) \bowtie E_3 \equiv E_1 \bowtie (E_2 \bowtie E_3)$$
$$(E_1 \underset{F_1}{\bowtie} E_2) \underset{F_2}{\bowtie} E_3 \equiv E_1 \underset{F_1}{\bowtie} (E_2 \underset{F_2}{\bowtie} E_3)$$

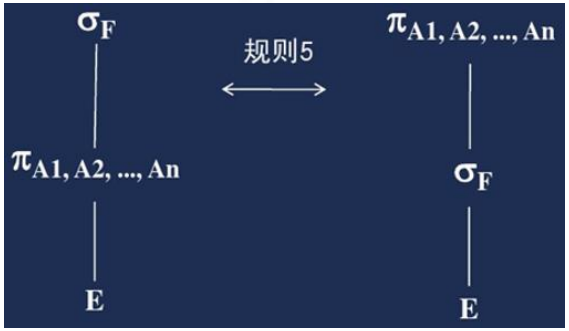
代数优化



– 3. 投影的串接定律

$$\pi_{A_1,A_2,\cdots,A_n}(\pi_{B_1,B_2,\cdots,B_m}(E)) \equiv \pi_{A_1,A_2,\cdots,A_n}(E)$$

其中{A1,A2,…,An}是{B1,B2,…,Bm}的子集



– 4. 选择的串接定律

$$\sigma_{F_1}(\sigma_{F_2}(E)) \equiv \sigma_{F_1 \wedge F_2}(E)$$

– 5. 选择与投影的交换律

$$\sigma_F(\pi_{A_1,A_2,\cdots,A_n}(E)) \equiv \pi_{A_1,A_2,\cdots,A_n}(\sigma_F(E))$$

– 6. 选择与笛卡尔积交换律

- F中只有E1的属性： $\sigma_F(E_1 \times E_2) \equiv \sigma_F(E_1) \times E_2$
- F=F1∧F2，且F1只有E1的属性、F2中只有E2的属性：

$$\sigma_F(E_1 \times E_2) \equiv \sigma_{F_1}(E_1) \times \sigma_{F_2}(E_2)$$

- F1只有E1的属性，F2有E1和E2的属性：

$$\sigma_F(E_1 \times E_2) \equiv \sigma_{F_2}(\sigma_{F_1}(E_1) \times E_2)$$

代数优化

- 7. 选择与并的分配律

$$\sigma_F(E_1 \cup E_2) \equiv \sigma_F(E_1) \cup \sigma_F(E_2)$$

- 8. 选择与差的分配律

$$\sigma_F(E_1 - E_2) \equiv \sigma_F(E_1) - \sigma_F(E_2)$$

- 9. 选择对自然连接的分配律

$$\sigma_F(E_1 \bowtie E_2) \equiv \sigma_F(E_1) \bowtie \sigma_F(E_2)$$

- 10. 投影与笛卡尔积的分配律

$$\pi_{A_1,A_2,\cdots,A_n,B_1,B_2,\cdots,B_m}(E_1 \times E_2) \equiv \pi_{A_1,A_2,\cdots,A_n}(E_1) \times \pi_{B_1,B_2,\cdots,B_m}(E_2)$$

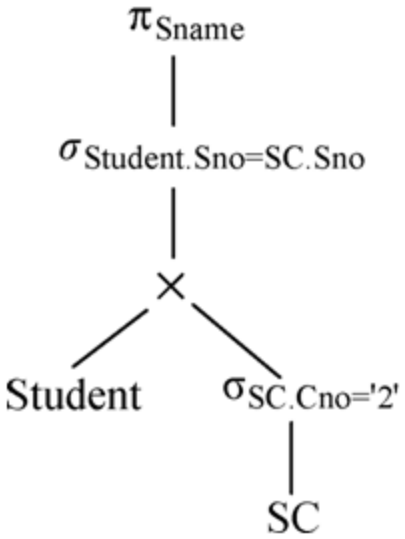
- 11. 投影与并的分配律

$$\pi_{A_1,A_2,\cdots,A_n}(E_1 \cup E_2) \equiv \pi_{A_1,A_2,\cdots,A_n}(E_1) \cup \pi_{A_1,A_2,\cdots,A_n}(E_2)$$

代数优化

- 查询树的启发式优化（优化的一般准则）
  - 选择运算尽早执行。是优化策略中最重要、最基本的一条。（减小中间关系-减少元组数据）
  - 投影运算尽早执行。（减小中间关系-减少属性数目）
  - 把投影运算和选择运算同时进行；把投影同其前或其后的双目运算结合起来。（减少扫描关系的次数）
  - 把某些选择同在它前面要执行的笛卡尔积结合起来成为一个连接运算。（把笛卡尔积与选择转换为连接）
  - 找出公共子表达式，把公共子表达式的结果写入中间文件，重复使用。（中间结果复用）

查询树的优化



- 查询树是关系代数表达式的树形表示
  - 操作的关系位于叶节点
  - 关系运算位于内部节点
  - 执行方式：自底向上执行，当一个内部节点的操作分量可用时，该节点的操作启动执行，执行结束后用结果关系代替该节点
- 查询树的构造
  - 将SQL语句转换为关系代数表达式：

    SELECT子句对应投影操作，FROM子句对应笛卡尔积，WHERE子句对应选择操作
  - 将关系代数表达式转换为查询树

查询树的优化

- 查询树优化算法
  - 利用规则4分解选择运算，把形如 $\sigma_{F_1 \wedge F_2}(E)$ 变换为 $\sigma_{F_1}(\sigma_{F_2}(E))$
  - 利用规则4-9把选择运算尽量移到叶端。
  - 利用规则3，5，10，11把投影运算尽量移到叶端
  - 利用规则3-5把选择和投影的串接合并成单个选择、单个投影或一个选择后跟一个投影的形式。使尽可能多的选择和投影同时执行

部分等价变换规则

- 3. 投影的串接定律

$$\pi_{A_1,A_2,\cdots,A_n}(\pi_{B_1,B_2,\cdots,B_m}(E)) \equiv \pi_{A_1,A_2,\cdots,A_n}(E)$$

其中{A1,A2,...,An}是{B1,B2,...,Bm}的子集

- 4. 选择的串接定律

$$\sigma_{F_1}(\sigma_{F_2}(E)) \equiv \sigma_{F_1 \wedge F_2}(E)$$

- 5. 选择与投影的交换律

$$\sigma_F(\pi_{A_1,A_2,\cdots,A_n}(E)) \equiv \pi_{A_1,A_2,\cdots,A_n}(\sigma_F(E))$$

- 6. 选择与笛卡尔积交换律
  - F中只有E1的属性： $\sigma_F(E_1 \times E_2) \equiv \sigma_F(E_1) \times E_2$
  - F=F1∧F2，且F1只有E1的属性、F2中只有E2的属性：

$$\sigma_F(E_1 \times E_2) \equiv \sigma_{F_1}(E_1) \times \sigma_{F_2}(E_2)$$

  - F1只有E1的属性，F2有E1和E2的属性：

$$\sigma_F(E_1 \times E_2) \equiv \sigma_{F_2}(\sigma_{F_1}(E_1) \times E_2)$$

- 7. 选择与并的分配律

$$\sigma_F(E_1 \cup E_2) \equiv \sigma_F(E_1) \cup \sigma_F(E_2)$$

- 8. 选择与差的分配律

$$\sigma_F(E_1 - E_2) \equiv \sigma_F(E_1) - \sigma_F(E_2)$$

- 9. 选择对自然连接的分配律

$$\sigma_F(E_1 \bowtie E_2) \equiv \sigma_F(E_1) \bowtie \sigma_F(E_2)$$

- 10. 投影与笛卡尔积的分配律

$$\pi_{A_1,A_2,\cdots,A_n,B_1,B_2,\cdots,B_m}(E_1 \times E_2) \equiv \pi_{A_1,A_2,\cdots,A_n}(E_1) \times \pi_{B_1,B_2,\cdots,B_m}(E_2)$$

- 11. 投影与并的分配律

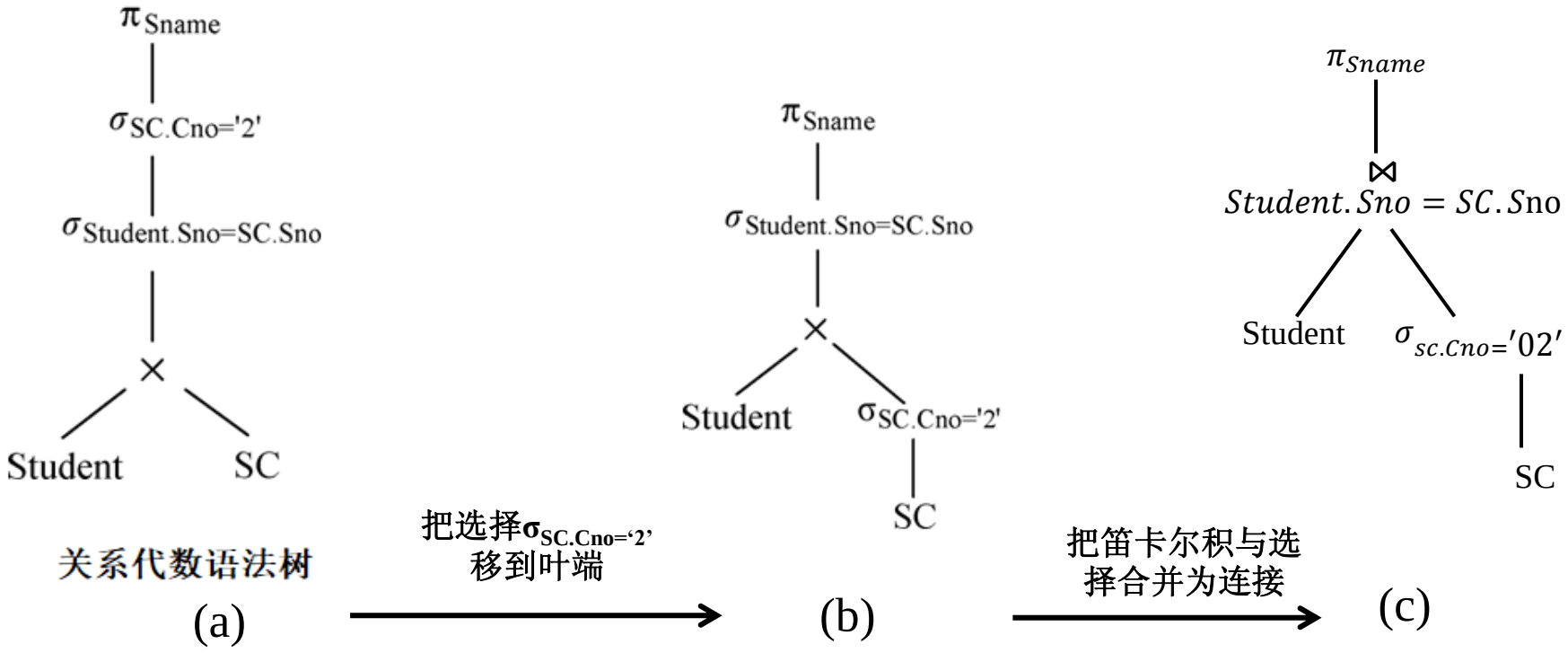
$$\pi_{A_1,A_2,\cdots,A_n}(E_1 \cup E_2) \equiv \pi_{A_1,A_2,\cdots,A_n}(E_1) \cup \pi_{A_1,A_2,\cdots,A_n}(E_2)$$

查询树的优化

- 查询树优化算法（续）
  - 对每个叶节点加必要的投影操作，以去掉对查询无用的属性
  - 如果笛卡尔积后还需按连接条件进行选择操作，可将两者组合成连接操作

示例1

- SELECT Student.Sname
  FROM Student，SC
  WHERE Student.Sno=SC.Sno AND SC.Cno=‘2’；



示例2

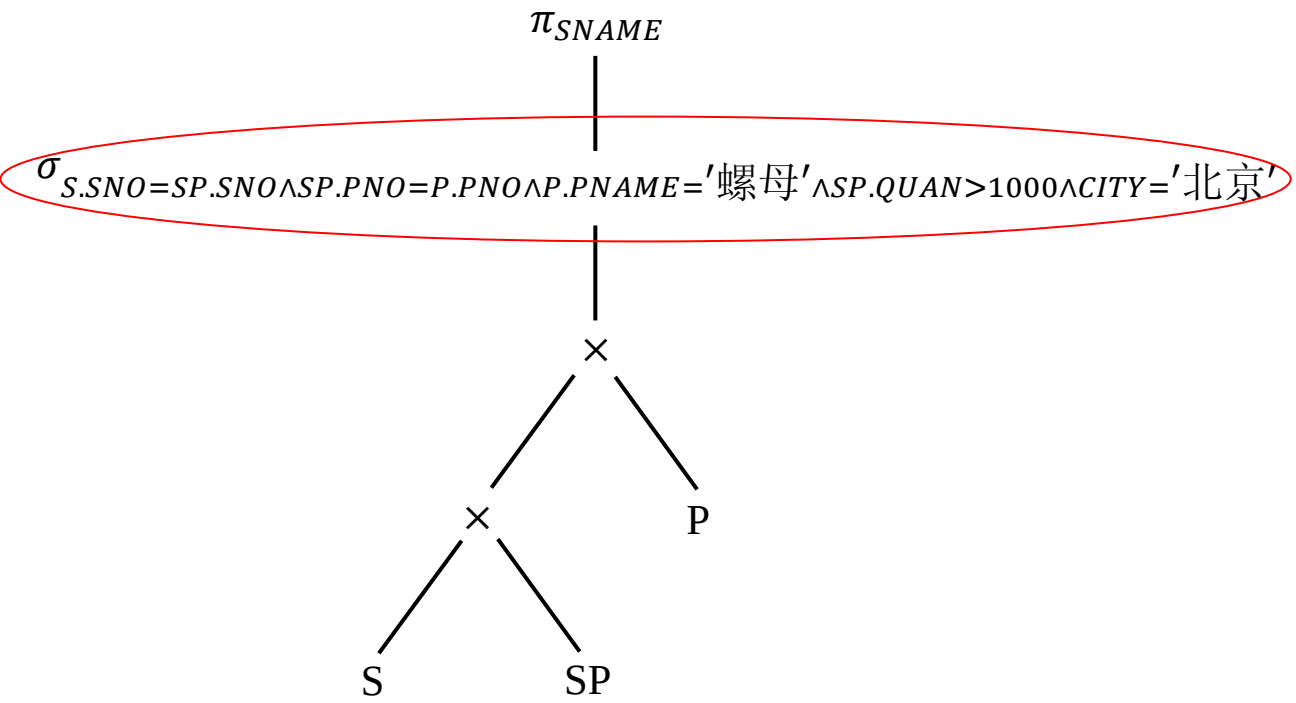
- 设有S（供应商），P（零件），SP（供应关系）：
  - S (SNO, SNAME,CITY), 供应商编号，供应商名称，供应商所在城市
  - P(PNO, PNAME)，零件编号，零件名称
  - SP(SNO, PNO, QUAN)，供应商编号，零件编号，数量

- 查询：北京供应商供应螺母数量大于1000的供应商名称。
  - SELECT SNAME
  - FROM S,P,SP
  - WHERE S.SNO= SP.SNO AND SP.PNO=P.PNO AND P.PNAME='螺母' AND SP.QUAN>1000;

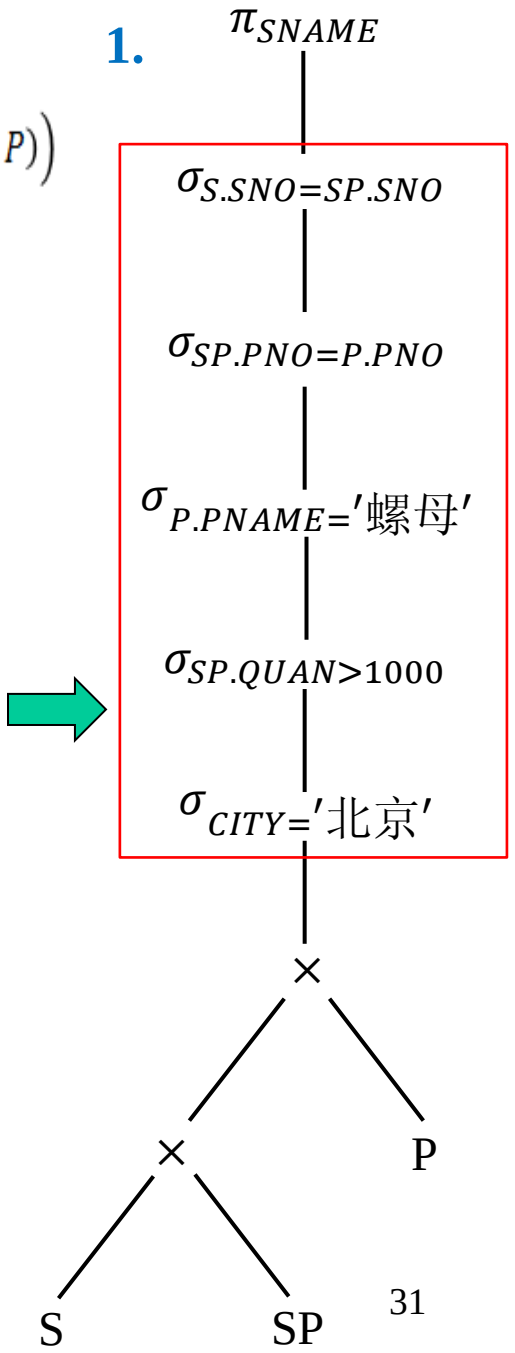
关系代数表达式：

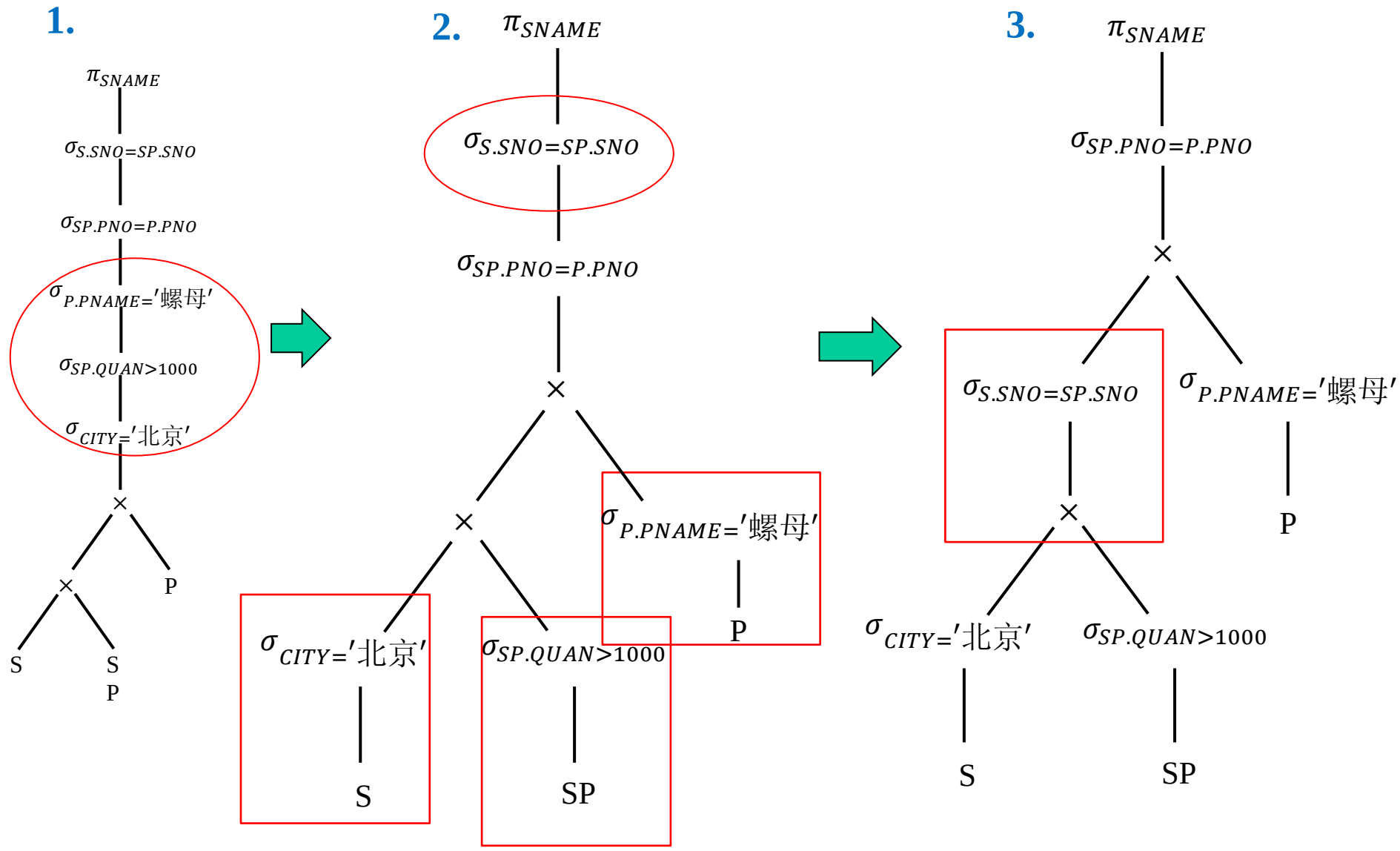
$$\pi_{SNAME}\left(\sigma_{S.SNO=SP.SNO\wedge SP.PNO=P.PNO\wedge P.PNAME='螺母'\wedge SP.QUAN>1000\wedge CITY='北京'}(S\times SP\times P)\right)$$

初始查询树：

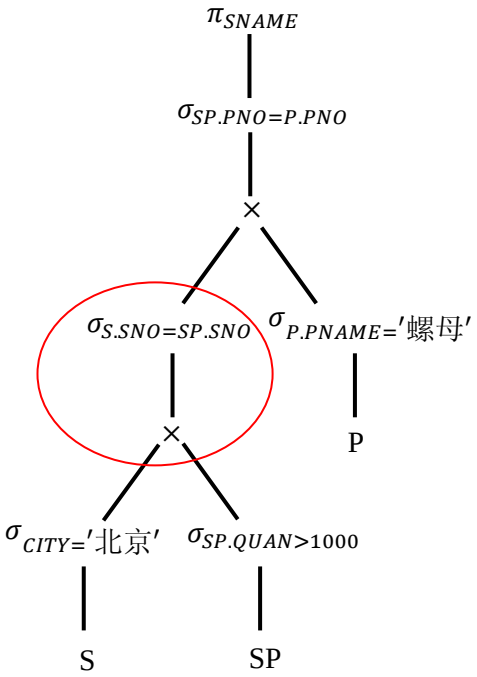


1.

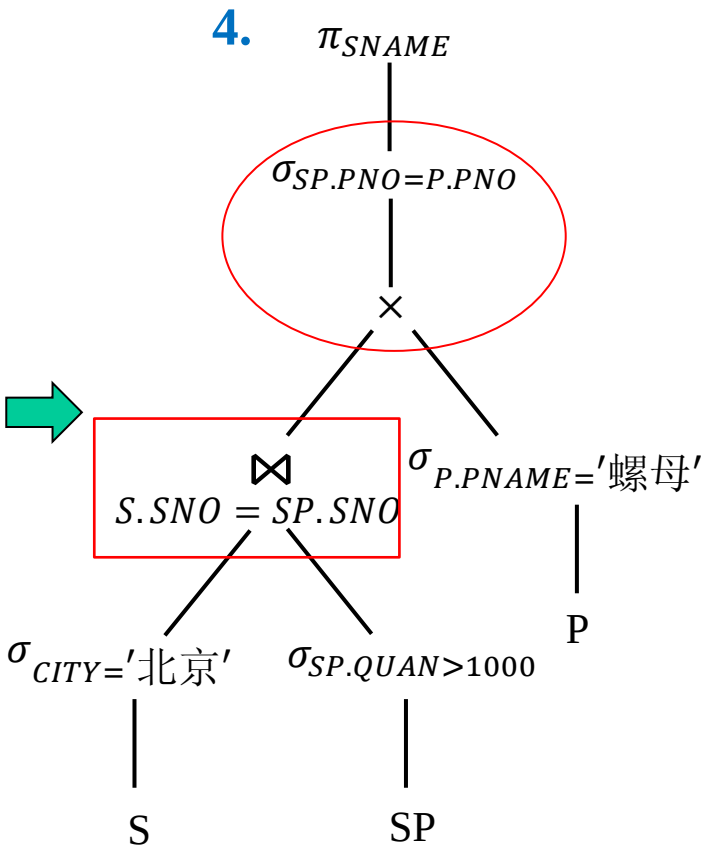



1.
π_{SNAME}
$\sigma_{S.SNO=SP.SNO}$
$\sigma_{SP.PNO=P.PNO}$
$\sigma_{P.PNAME='螺母'}$
$\sigma_{SP.QUAN>1000}$
$\sigma_{CITY='北京'}$
×
×
P
S
S
P
2.
π_{SNAME}
$\sigma_{S.SNO=SP.SNO}$
$\sigma_{SP.PNO=P.PNO}$
×
×
$\sigma_{P.PNAME='螺母'}$
P
$\sigma_{CITY='北京'}$
S
$\sigma_{SP.QUAN>1000}$
SP
3.
π_{SNAME}
$\sigma_{SP.PNO=P.PNO}$
×
$\sigma_{S.SNO=SP.SNO}$
$\sigma_{P.PNAME='螺母'}$
×
P
$\sigma_{CITY='北京'}$
S
$\sigma_{SP.QUAN>1000}$
SP

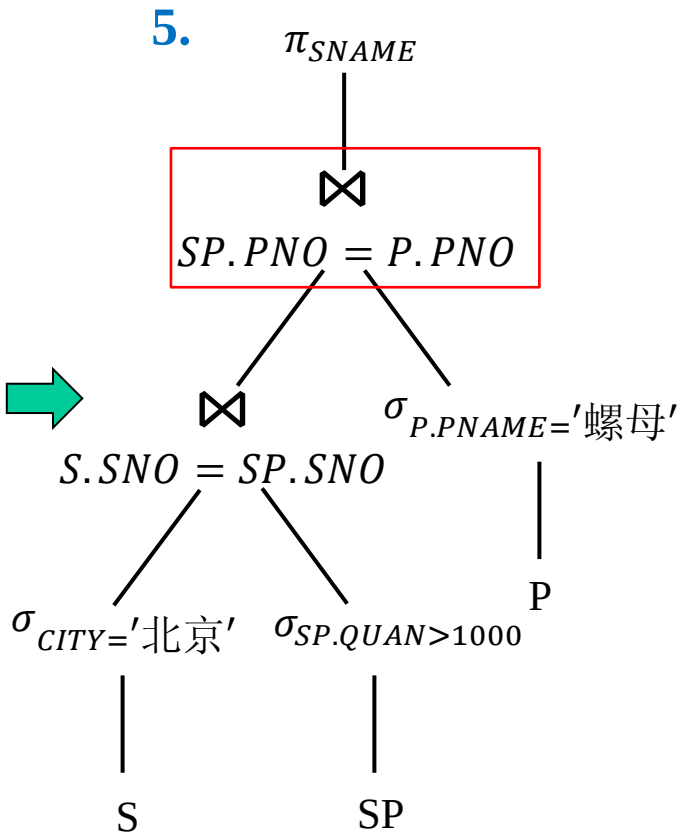
3.


π_{SNAME}
$\sigma_{SP.PNO=P.PNO}$
×
$\sigma_{S.SNO=SP.SNO}$
$\sigma_{P.PNAME='螺母'}$
×
P
$\sigma_{CITY='北京'}$
$\sigma_{SP.QUAN>1000}$
S
SP

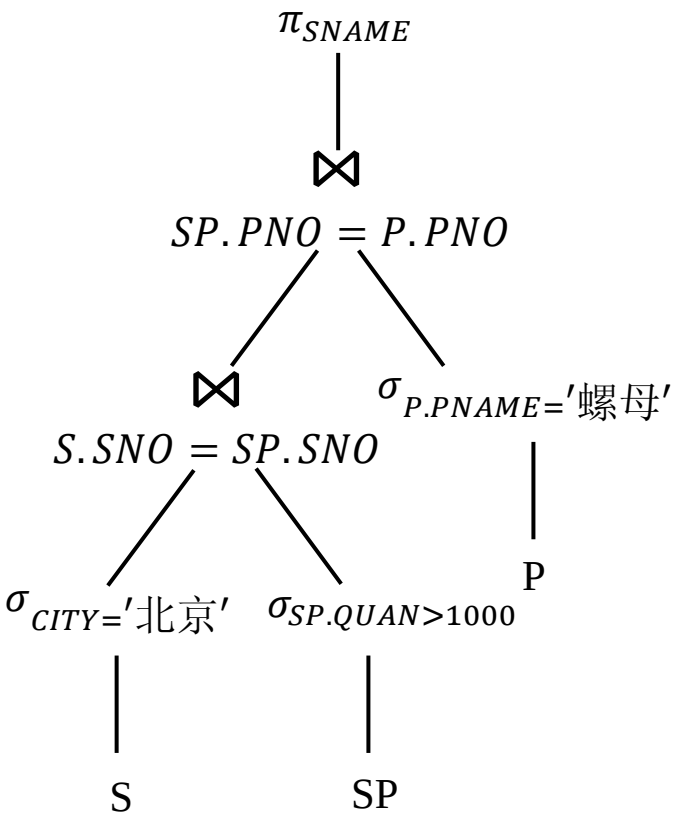
4.


π_{SNAME}
$\sigma_{SP.PNO=P.PNO}$
×
⋈
$S.SNO = SP.SNO$
$\sigma_{P.PNAME='螺母'}$
P
$\sigma_{CITY='北京'}$
$\sigma_{SP.QUAN>1000}$
S
SP

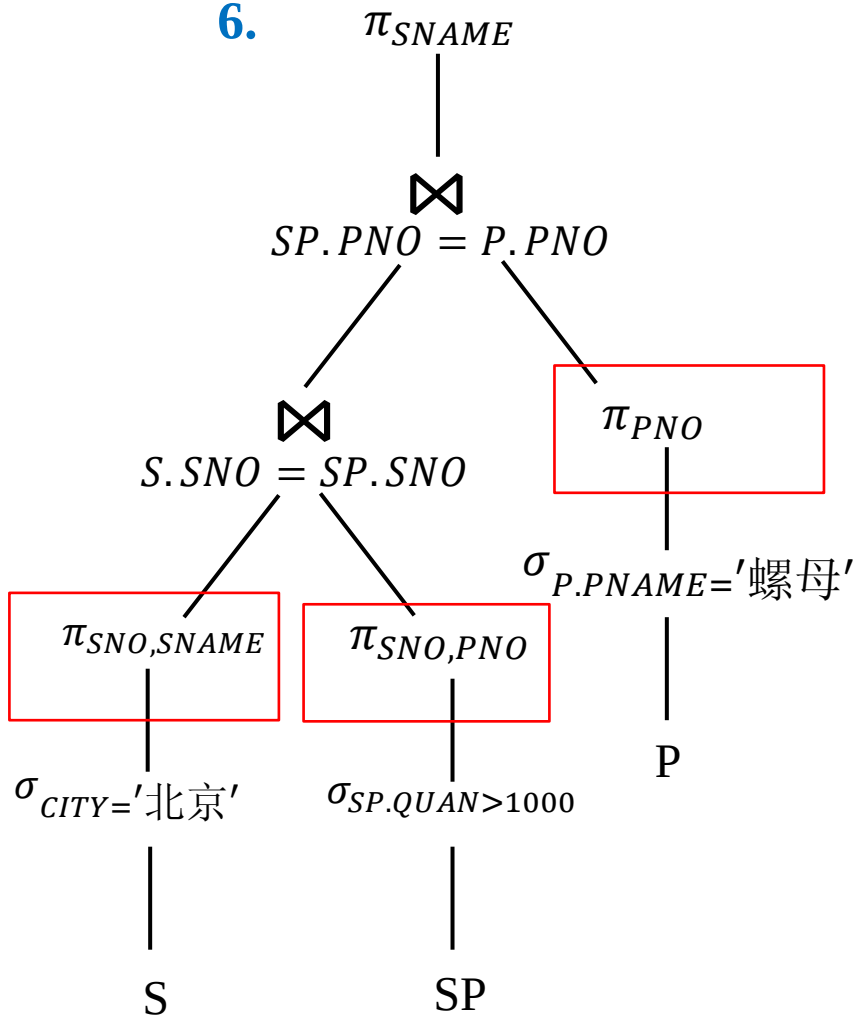
5.


π_{SNAME}
⋈
$SP.PNO = P.PNO$
⋈
$S.SNO = SP.SNO$
$\sigma_{P.PNAME='螺母'}$
P
$\sigma_{CITY='北京'}$
$\sigma_{SP.QUAN>1000}$
S
SP

5.


π_{SNAME}
$SP.PNO = P.PNO$
$S.SNO = SP.SNO$
$\sigma_{P.PNAME='螺母'}$
$\sigma_{CITY='北京'}$
$\sigma_{SP.QUAN>1000}$
P
S
SP

6.


π_{SNAME}
$SP.PNO = P.PNO$
$S.SNO = SP.SNO$
π_{PNO}
$\sigma_{P.PNAME='螺母'}$
$\pi_{SNO,SNAME}$
$\pi_{SNO,PNO}$
P
$\sigma_{CITY='北京'}$
$\sigma_{SP.QUAN>1000}$
S
SP

查询树的优化

- 分组：双目运算和它的直系祖先(选取、投影)为一组；双目运算后代直到叶子全是单目运算时并入该组。笛卡尔积的后面若不是与之可以合并的自然连接的等值选择时，其后代单独分为一组。
- 生成查询代码时，每组节点的计算是程序中的一步。各步的顺序是任意的，只要保证任何一组的计算不会在其后代组之前计算。

物理优化

- 选择高效合理的操作算法或存取路径，得到优化的查询计划。
- 常用方法
  - 基于规则的启发式优化方法
  - 基于代价估算的优化方法
  - 两者结合的优化方法

基于启发式规则的存取路径选择优化

- 选择操作的启发式规则
  - 对于小关系，使用全表顺序扫描。
  - 对于大关系：可以采用索引扫描法（如结果的元组数目较小），全表顺序扫描。
- 连接操作的启发式规则
  - 如果两个表都已经按照连接属性排序——排序-合并法；
  - 如果一个表在连接属性上有索引——索引连接法；
  - 如果连接属性上未排序且未建索引，且其中一个表较小——Hash join法；
  - 最后可选用嵌套循环法，并选择较小的表为外循环表。

基于代价估算的优化

- 利用数据库的统计信息计算各种操作算法的执行代价，选出具有最小代价的执行计划
- 数据库统计信息主要包括：
  - 每个基本表的规模，包括元组数、元组长度、占用块数以及溢出块数等
  - 基本表每个列的信息，包括不同值个数、最大与最小值，是否有索引以及索引类型等
  - 索引的具体信息，例如B+树索引的层数、不同索引值个数等

查询优化的一般步骤

- 把查询转换成语法树，如关系代数语法树。
- 把语法树利用代数优化转换成优化后的标准形式。
- 利用基于启发式规则的物理优化，选择底层的存取路径。生成查询计划，利用基于代价的物理优化，选择代价最小的。

小结

- 查询处理的步骤
- 查询操作的实现算法
- 查询的代数优化和物理优化的概念和原则
- 查询优化的一般步骤