空间数据库2018秋季

查询处理与优化-1227

陈斌

北京大学地球与空间科学学院

gischen@pku.edu.cn

查询处理与优化

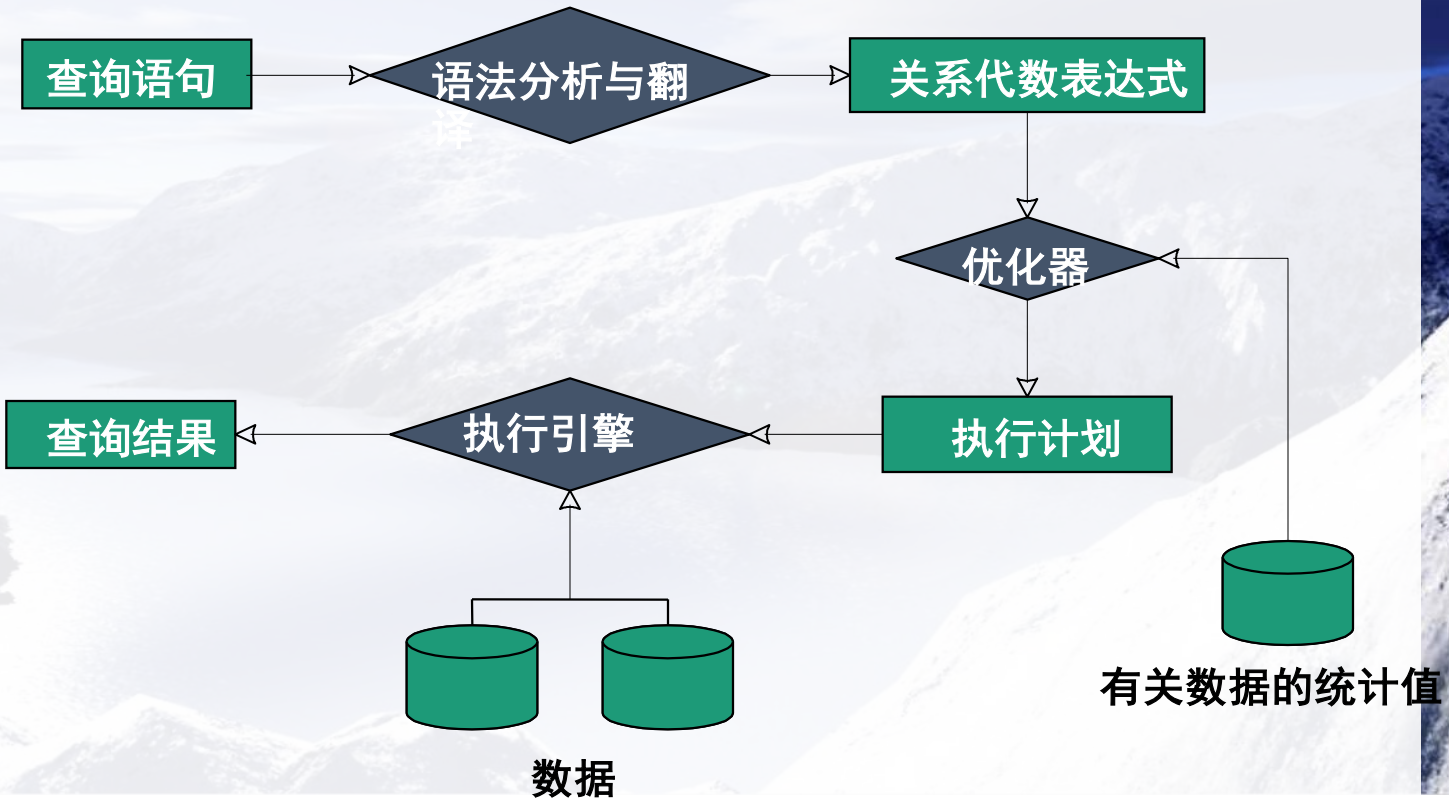
- 查询处理概览
- 代价估算
- 基本运算的实现与代价
- 关系代数表达式实现
- 关系代数表达式转换
- 选择执行计划
- 空间查询处理与优化

查询处理概览

- 查询处理是指从数据库中提取数据的一系列活动。主要包括：
  - 将用高层数据库语言表示的查询语句翻译为能在文件系统这一物理层次上实现的表达式
  - 为优化查询而进行各种转换
  - 查询的实际执行
- 输入：SQL语句；
- 输出：满足查询条件的数据

查询处理基本步骤

1. 语法分析与翻译
2. 优化
3. 执行



查询优化概念

- 查询优化是为关系代数表达式的计算选择<u>最有效的查询计划</u>的过程
- 查询执行计划：用于计算查询的原语序列
- 执行原语：加了“如何执行”注释的关系代数运算（选择、投影……）
  - 根据选择的算法对文件记录进行操作

查询优化过程

- 代数优化
  - 力图找出与给定关系代数表达式等价的但执行效率更高的一个表达式
- 执行策略选择
  - 对查询语句处理的详细策略的选择
    - 选择执行运算所采用的具体算法
    - 选择将使用的特定索引等等

查询优化过程

- 可能性
  - SQL语言和关系代数表达式的非过程化特点
- 可行性
  - 查询优化器具有丰富的可使用信息
  - 数据库发生变化时优化器容易再次进行优化
  - 优化器能够对多种实现策略逐一进行考虑
  - 优化器集中了最优秀的程序员的智慧和经验

代价估算：关系统计信息

- n_r：关系r中的元组数目(number)
- b_r：含有关系r的元组的块数目(block)
- s_r：关系r中一个元组的大小(size)
- f_r：关系r的块因子，即一个块中能存放的关系r的元组数(factor)
- 若假定关系r的元组物理上存于同一文件中，则：$b_r = \mathrm{Roof}(n_r / f_r)$

代价估算：关系统计信息

- V(A,r)：关系r中属性A所具有的不同值的数目。
  - V(A,r)等于$\Pi_A(r)$的大小
  - 若A为关系r的码，则$V(A,r) = n_r$
- SC(A,r)：关系r的属性A的选择基数。表示关系r中满足属性A上的一个等值条件的平均元组数。
  - 若A为r的码属性，则SC(A,r) ＝1
  - 若A为非码属性，并假定V(A,r)个不同的值在元组上均匀分布，则$SC(A,r) = (n_r / V(A,r))$。
- 说明：V(A,r)与SC(A,r)中的A可以是属性组。

代价估算：索引统计信息

- f_i： 树形结构(如B+树)索引i的内部结点的平均扇出。
- HT_i： 索引i的层数。
  - 对于关系r的属性A上所建的平衡树索引（如B+树），$HT_i = Roof(\log_{f_i}(V(A,r)))$
  - 对于散列索引，$HT_i = 1$ □
- LB_i： 索引i中最低层索引块数目，即索引叶层的块数。
  - 对于散列索引，LB_i就是索引中的块数。
- 算法A的代价估计记为E_A

查询代价度量

- 查询代价：查询处理对各种资源的使用情况
  - 磁盘存取（简化为磁盘块传送数）
  - CPU时间
  - 通信开销
- 一个重要的影响因素：主存中缓冲区的大小M
  - 最好的情形，所有的数据可以读入到缓冲区中
  - 最坏的情形，缓冲区只能容纳数目不多的数据块——大约每个关系一块。

基本运算的实现与代价

- 每个基本关系代数运算都有多种实现算法
  - 适合不同的情况
    - 等值条件 vs 范围条件
    - 数据是聚集 vs 非聚集
    - 相关属性上有索引 vs 没有索引
  - 具有不同的执行代价
- 选择、排序、连接、其它运算

选择运算：全表扫描

- 方法：依次访问表的每一个块，对于每一个元组，测试它是否满足选择条件。
- 代价：$E = b_r$
- 缺点：效率低
- 优点：
  - 对关系的存储方式没有要求，不需要索引。
  - 适用于任何选择条件。

选择运算：索引扫描

- 条件：表在选择条件的属性上建有索引。
- 方法：访问索引，根据索引项的指示去访问数据元组。
  - 无序索引：访问满足等值条件的元组
  - 有序索引：访问满足范围查找条件的一系列元组。

选择运算：索引扫描代价

- 利用主索引，等值比较：
  - $E = HT_i + Roof(SC(A,r) / f_r)$
- 利用辅助索引，等值比较：
  - $E = HT_i + SC(A,r)$
- 利用主索引，非等值比较：
  - $E = HT_i + b_r/2$
    - （假设大约一半的元组满足比较条件）
- 利用辅助索引，非等值比较：
  - $E = HT_i + LB_i/2 + n_r/2$

选择运算：复杂条件

- 复杂条件的选择
  - 合取：$\sigma_{\theta 1 \wedge \theta 2 \wedge \ldots \wedge \theta n}(r)$
  - 析取：$\sigma_{\theta 1 \vee \theta 2 \vee \ldots \vee \theta n}(r)$
- 方法
  - 利用一个索引进行合取选择。
  - 利用组合索引进行合取选择。
  - 利用多维索引进行合取选择。
  - 通过标识符的交集进行合取选择。
  - 通过标识符的并集进行析取选择。

排序运算

- 排序的必要性
  - SQL查询可能要求有序的查询结果。
  - 事先对于作为运算对象的关系进行排序，可以提高某些关系运算（例如连接）的执行效率。
- 内排序：文件较小，整个排序过程都能在内存中进行
  - 许多成熟的算法
- 外排序：文件较大，排序过程需要使用外存。
  - 以内排序为基础

外排序：归并算法

- 设内存中用于排序的缓冲区页面数为M
- 第一阶段，建立多个已排序的子表。
  - 每次读入M块，进行内排序，将长度为M块的已排序子表（共Roof(br /M)个）写到外存中。
- 第二阶段，对已排序子表进行归并（可能需进行若干趟）。

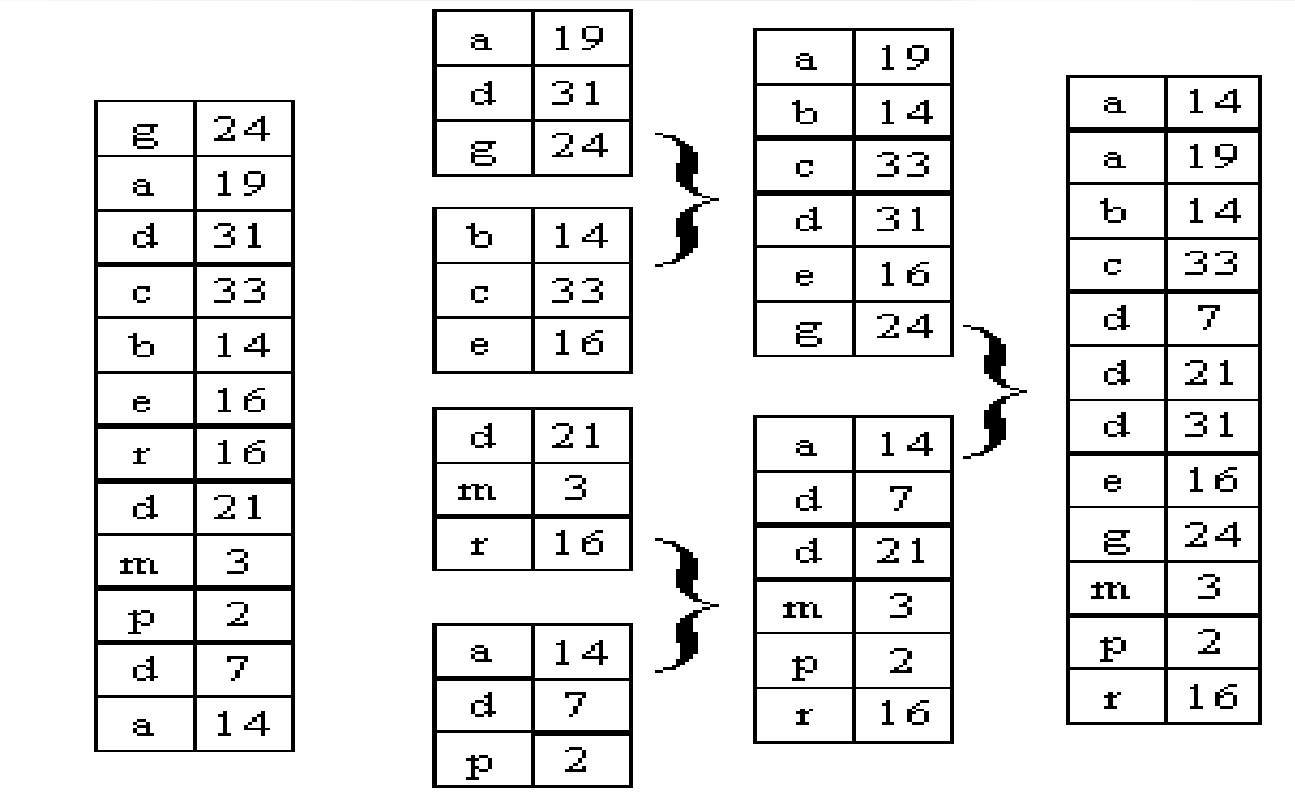
外排序：归并算法第二阶段

- 第一趟：将头M-1个已排序子表的各块逐步读入内存，归并并输出。
  - 将下M-1个已排序子表的各块逐步读入内存，归并并输出。
  - ……
  - 已排序子表数目减少到原来的1/（M-1）
- 第二趟：以第一趟的输出作为输入，重复过程。
- 第三趟：以第二趟的输出作为输入，重复归并过程
- ……直至归并为一个已排序文件。

外排序：归并算法归并过程

- 将头M-1个已排序子表的每个的第一块读入内存的一个缓冲页
  - repeat
    - 在所有缓冲页中第一个元组中挑选排序码值最小的元组；
    - 把该元组写到第M缓冲页，将其从原缓冲页中删除；
    - if 任何一个归并段文件Ri的缓冲页为空且没有到达Ri末尾
      - then 读入Ri的下一块到相应的缓冲页；
    - if 第M个缓冲页满
      - then 将第M个缓冲页写到磁盘，并清空该缓冲页；
  - until 所有的缓冲页均空
- 将下M-1个已排序子表的每一个的第一块读入内存，归并。
  - …….

外排序：归并算法



初始关系　　归并段文件　　归并段文件　　排序结果

第一阶段
创建归并段文件

第二阶段
第一趟归并

第二阶段
第二趟归并

外排序：归并算法代价估算

- 趟数 = Roof($\log_{M-1}$(b_r /M))
- E = b_r (2*Roof($\log_{M-1}$(br /M)) + 1)
  - 第一阶段：br
  - 第二阶段：br*2*趟数（每趟的读+写）

连接运算

- 二元运算，涉及两个关系及连接条件
  - 条件连接：$r \bowtie_{\theta} s$
  - 自然连接：$r \bowtie s$
- 连接运算是非常重要的运算，有多种实现算法

连接运算

- 自然连接结果集大小的估计：
- 基于主码、外码连接的情况：结果集的元组数等于外码所在关系的元组数。
- 一般情况：（假设连接属性A的每个值在关系的元组中等概率出现），结果集的元组数为：
  - $n_r*(n_s/V(A,s))$ 或 $n_s*(n_r/V(A,r))$
  - 当V(A,s)与V(A,r)不同时，取两个估计值中较小者

连接运算：实现算法

- 嵌套循环连接
- 块嵌套循环连接
- 索引嵌套循环连接
- 排序-归并连接
- 散列连接
- 复杂连接的实现

连接运算：嵌套循环

- 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，把$t_r \square t_s$加到结果中
  - end
- end

连接运算：嵌套循环

- 优点：对参加运算的关系没有要求，适合于任何连接条件。
- 代价：
  - 最坏情况（缓冲区只能够容纳每个关系的一个块）：
    - $n_r*b_s + b_r$ 或 $n_s*b_r + b_s$
  - 最好情况（内层关系s能完全放在内存中）：
    - $b_s + b_r$

连接运算：块嵌套循环

- 块嵌套循环连接：以块的方式循环，以减少块读写次数
  - for each 块B_r of r do begin
  - for each 块B_s of s do begin
  - for each 元组t_r in Br do begin
  - for each 元组t_s in Bs do begin
    - 测试元组对(t_r,t_s)是否满足连接条件θ
    - 如果满足，把$t_r \square t_s$加到结果中
  - end end end end

连接运算：块嵌套循环

- 优点：对参加运算的关系没有要求，适合于任何连接条件。
- 代价：
  - 最坏情况（缓冲区只能够容纳每个关系的一个块）：
    - $b_r * b_s + b_r$ 或 $b_s * b_r + b_s$
  - 最好情况（内层关系s能完全放在内存中）：
    - $b_s + b_r$

连接运算：块嵌套循环

- 一些改进
  - 连接属性是内层关系的码时，内层循环可中途跳出。
  - 内层循环轮流做正、反向扫描，重用缓冲区中的数据，以减少磁盘读取。
  - 内层循环利用索引。

连接运算：索引嵌套循环

- 索引嵌套循环连接：在内层循环中利用连接属性上的索引
- for each 元组 t_r in r
- do begin
  - for each 与元组t_r满足连接条件的索引项 in s关系在连接属性上的索引
  - do begin
    - 利用索引取到相应的t_s
    - 把t_r □ t_s加到结果中
  - end
- end

连接运算：索引嵌套循环

- 代价：
  - 最坏情况（缓冲区只能够容纳关系r的一个块和索引的一个块）：
    - $b_r + \underline{n_r * c}$ 或 $b_s + \underline{n_s * c}$

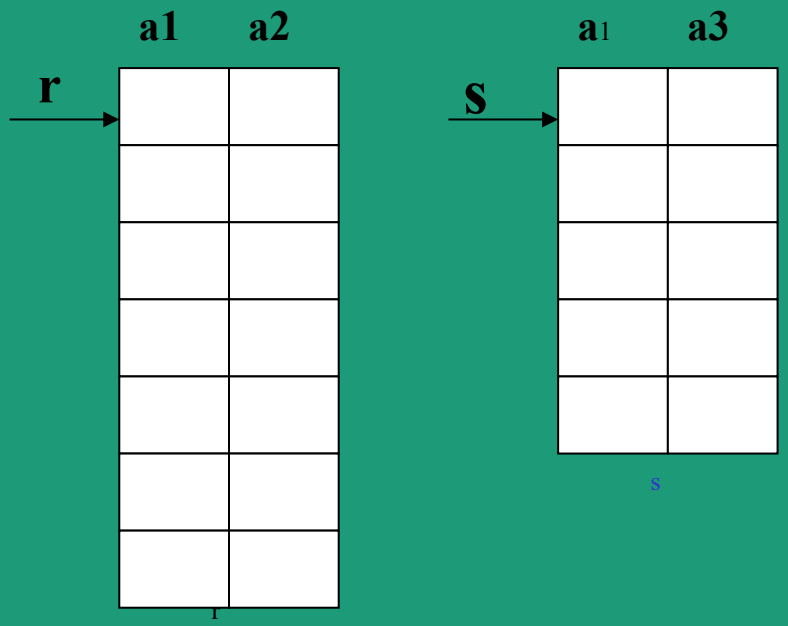
对关系S使用连接条件利用索引进行选择操作的代价

  - 最好情况不必考虑，因为不必采用索引嵌套循环连接方法。

连接运算：排序-归并

- 排序-归并连接
  - 类似于外排序的归并算法的思路，进行连接运算。
  - 前提：两个关系的元组都已按连接属性排好序。
  - 适用于自然连接和等值连接。

连接运算：排序-归并



在归并连接中使用的已排序关系

连接运算：排序-归并


```
pr := r的第一个元组的地址
ps := s的第一个元组的地址；
while (ps≠null and pr≠null) do
begin  t_s := ps所指向的元组；    S_s := { t_s } ；
          让ps指向关系s的下一个元组；    done := false;
          while (not done and ps≠null) do
       begin  t_s′ := ps所指向的元组；
            if (t_s′ [JoinAttrs] = t_s[JoinAttrs])
              then begin   S_s := S_s∪{ t_s′ } ；
                                 让ps指向关系s的下一个元组；
                     end
              else  done := true ；
       end
```

在连接属性上具有相同值的S元组被加入到了S_s中。Ps指向在连接属性上具有另一个值的S元组。

连接运算：排序-归并


```
t_r := pr所指向的元组；
    while (pr≠null and t_r[JoinAttrs] < t_s[JoinAttrs]) do
      begin  让pr指向关系r的下一个元组；
             t_r := pr所指向的元组；
      end
    while (pr≠null and t_r[JoinAttrs] = t_s[JoinAttrs]) do
             begin
        for each t_s in S_s do
             begin  将t_s t_r加入结果中；end
          让pr指向关系r的下一个元组；
             t_r := pr所指向的元组；
             end
end
```

在r中跳过中不能与S_s中的s元组匹配的r元组。

在r中前进，将r元组与S_s中的每个s元组连接，直至r元组中的连接属性值大于s元组的连接属性值。

连接运算：排序-归并

- 在归并连接中使用的已排序关系
- 代价：
  - $b_r + b_s$

pr →

a1	a2
a	3
b	1
d	8
d	13
f	7
m	5
q	6

r

ps →

a1	a3
a	A
a	G
c	L
d	N
m	B

s

在归并连接中使用的已排序关系

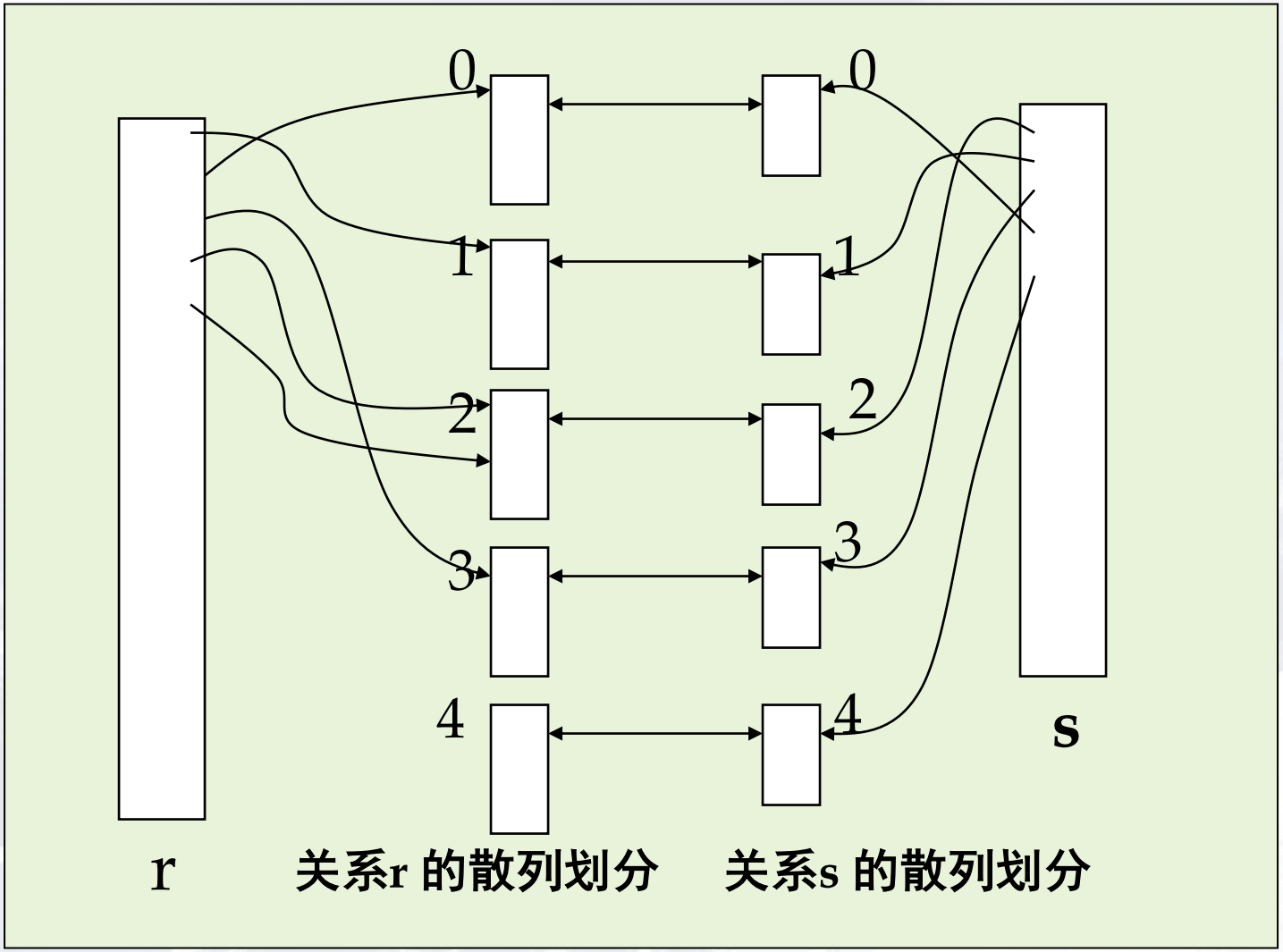
连接运算：散列连接

- 适用于自然连接和等值连接。
  - 非等值/范围
- 基本思想
  - 将两个关系按连接属性值划分成有相同散列函数值的元组集合。关系r在一个散列划分中的元组只需要与关系s在对应的划分中的元组相比较。在r和s的每一对划分中进行索引嵌套循环连接（散列索引）。

连接运算：散列连接

- 方法：
  - 确定连接属性上的散列函数h，用于对s元组和r元组进行划分。
  - 确定连接属性上的散列函数h’，用于逐个对每一划分中的s元组构造散列索引，
  - 再对同一划分中的r元组查找散列索引，同时进行连接。

连接运算：散列连接



连接运算：复杂连接

- 合取式：$r \bowtie_{\theta 1 \wedge \theta 2 \wedge \ldots \wedge \theta n} s$

  用一种高效技术计算某一条件下的连接 $r \bowtie_{\theta 1} s$，在生成结果元组时测试其他条件。

- 析取式：$r \bowtie_{\theta 1 \vee \theta 2 \vee \ldots \vee \theta n} s$

  $\Longrightarrow r \bowtie_{\theta 1} s \cup r \bowtie_{\theta 2} s \ldots$

  $r \bowtie_{\theta n} s$

其它运算

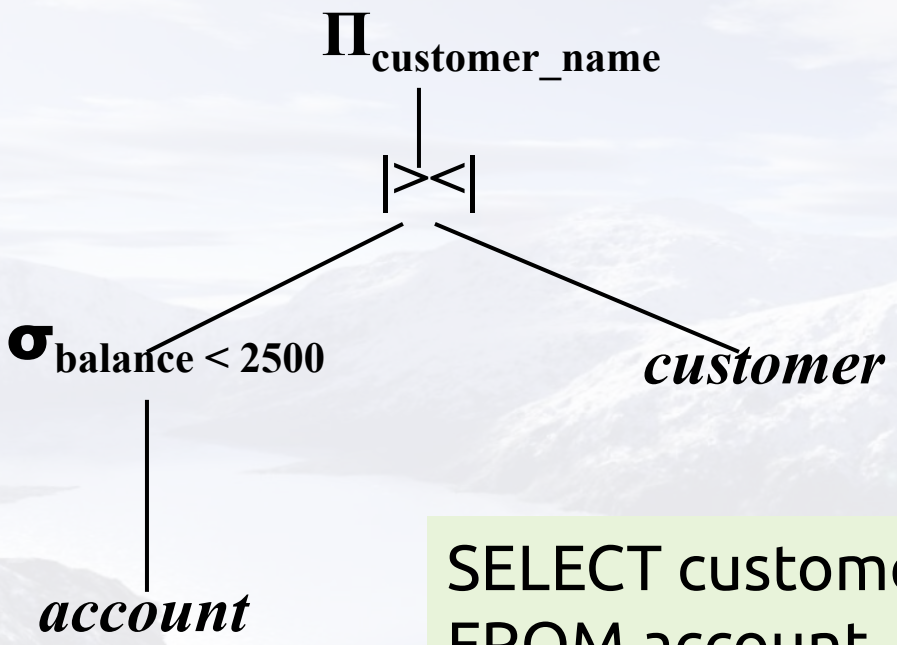
- 消除重复
  - 用排序的方法
  - 用散列的方法
- 投影
  - 投影，消除重复
- 并、交、差
  - 排序的方法
  - 散列的方法

其它运算

- 外连接
  - 计算连接，然后将适当的元组加到结果中。
  - 例 $r]><|_{\theta} s$
  - 计算 $r |><|_{\theta} s$ ==> q1
  - 计算 $r - \Pi_r(q1)$，在s对应的分量填上空值，加到q1中。
- 聚集
  - 排序的方法
  - 散列的方法

关系代数表达式实现

- $\Pi_{customer\_name}(\sigma_{balance < 2500}(account) |\bowtie| customer)$




```
SELECT customer.name
FROM account, customer
WHERE account.name=customer.name
AND account.balance<2500
```

关系代数表达式实现

- 实体化的方法
  - 建立临时关系
  - 代价：各个运算的代价加上中间结果写到磁盘的代价。(n_r/f_r)
- 流水线的方法
  - 不建临时关系，一个操作的结果传给下一个操作作为输入。

关系代数表达式实现

- 流水线的实现：
  - 需求驱动：在操作树的顶端将数据往上拉。
  - 生产者驱动： 将数据从操作树的底层往上推
- 需求驱动的流水线方法比生产者驱动的流水线方法使用更广泛，因为它更容易实现。

关系代数表达式实现

- 权衡：
  - 流水线技术限制了实现操作的可用算法。
  - 例：若连接运算的左端输入来自流水线，则不可用排序-归并连接算法，可用索引嵌套循环连接算法。
  - 若连接运算的两端输入均来自流水线，则限制更大。
- 并非所有情况下都是流水线方法的代价小于实体化方法的代价

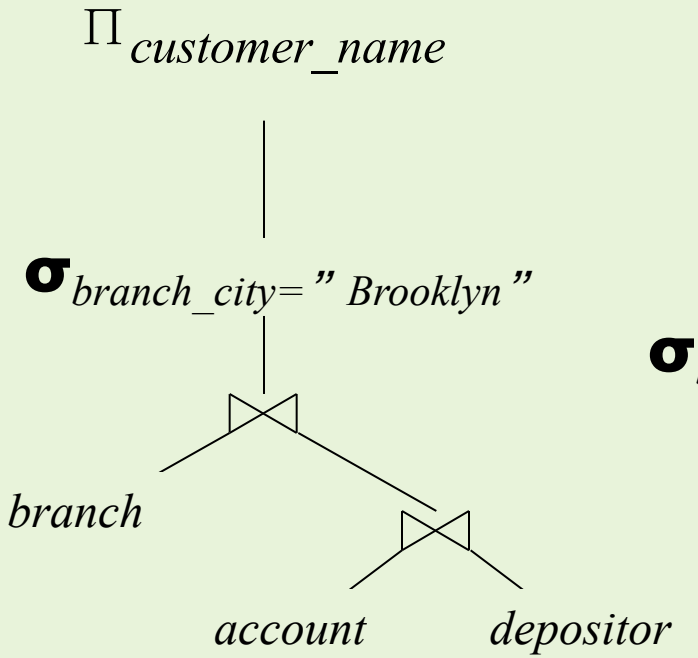
关系代数表达式转换

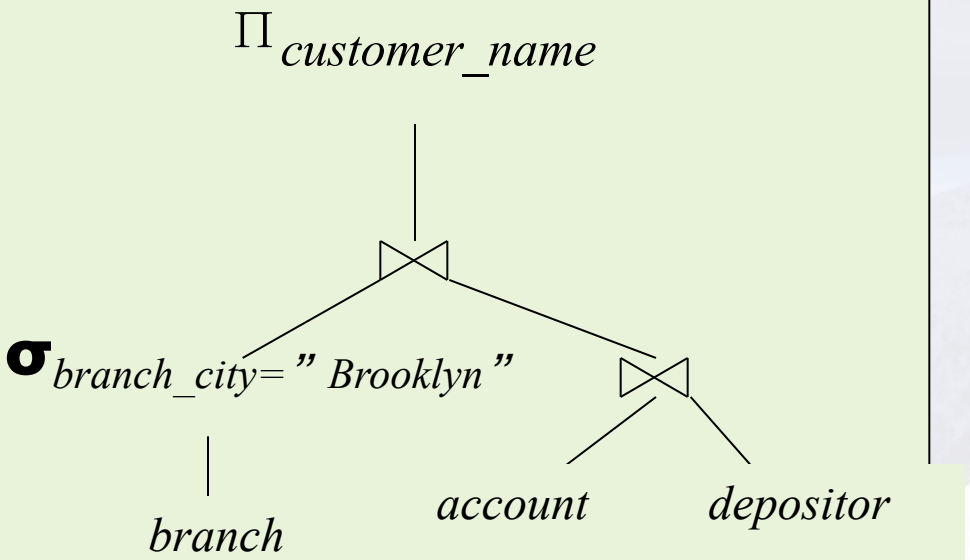
- 查询优化是为关系代数表达式的计算选择最有效的查询计划的过程。
- 查询优化的过程：
  - 代数优化：力图找出与给定关系代数表达式等价的但执行效率更高的一个表达式。
  - 查询语句处理的详细策略的选择，例如选择执行运算所采用的具体算法，选择将使用的特定索引等等。

等价表达式

- 两个表达式等价：产生的结果关系具有相同的属性集和相同的元组集。
- 例$\Pi_{customer_name}(\sigma_{branch\text{-}city=\text{"Brooklyn"}}(branch \bowtie (account \bowtie depositor)))$
- 等价于
- $\Pi_{customer\_name}((\sigma_{branch\text{-}city=\text{"Brooklyn"}}(branch)) \bowtie (account \bowtie depositor))$

等价表达式


Π customer_name
σ branch_city=" Brooklyn"
branch
account
depositor


Π customer_name
σ branch_city=" Brooklyn"
branch
account
depositor

参考文献

- [TP311.13/261]空间数据库 = Spatial databases a tour (美) Shashi Shekhar, Sanjay Chawla著 谢昆青 ... 等译 北京 机械工业出版社 2004
- 北京大学计算机系数据库教研室
  - 数据库原理与技术讲义（杨冬青）
- 数据库系统概论
  - 岳丽华，丁卫群 编著
  - 科学出版社，2000年