2024年春季学期

数据库系统概论

An Introduction to Database Systems

第九章 关系查询处理和查询优化

中国科学技术大学 大数据学院

黄振亚，huangzhy@ustc.edu.cn

9.1 关系数据库系统的查询处理

9.2 关系数据库系统的查询优化

9.3 代数优化

9.4 物理优化

9.5 小 结

9.3 代 数 优 化

- 9.3.1 关系代数表达式等价变换规则
- 9.3.2 查询树的启发式优化

9.3.1 关系代数表达式等价变换规则

- 代数优化策略：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
- 两个关系表达式E_1和E_2是等价的，可记为$E_1 \equiv E_2$

关系代数表达式等价变换规则（续）

- 常用的等价变换规则：

1. 连接、笛卡尔积交换律

设E_1和E_2是关系代数表达式，F是连接运算的条件，则有

$$E_1 \times E_2 \equiv E_2 \times E_1$$

$$E_1 \bowtie E_2 \equiv E_2 \bowtie E_1$$

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

2. 连接、笛卡尔积的结合律

设E_1，E_2，E_3是关系代数表达式，F_1和F_2是连接运算的条件，则有

$$(E_1 \times E_2) \times E_3 \equiv E_1 \times (E_2 \times E_3)$$

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

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

关系代数表达式等价变换规则（续）

3. 投影的串接定律

$$\prod_{A_1A_2\ldots A_n}(\prod_{B_1B_2\ldots B_m}(E)) \equiv \prod_{A_1A_2\ldots A_n}(E)$$

这里，E是关系代数表达式，$A_i(i=1，2，\ldots，n)$，$B_j(j=1，2，\ldots，m)$是属性名且$\{A_1，A_2，\ldots，A_n\}$构成$\{B_1，B_2，\ldots，B_m\}$的子集。

4. 选择的串接定律

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

这里，E是关系代数表达式，F_1、F_2是选择条件。

选择的串接律说明选择条件可以合并。这样一次就可检查全部条件。

关系代数表达式等价变换规则（续）

5. 选择与投影操作的交换律

$$\sigma_F(\prod_{A_1A_2\ldots A_n}(E)) \equiv \prod_{A_1A_2\ldots A_n}(\sigma_F(E))$$

- 选择条件F只涉及属性A_1，…，A_n

若F中有不属于A_1，…，A_n的属性B_1，…，B_m则有更一般的规则：

$$\prod_{A_1, A_2, \ldots, A_n}(\sigma_F(E)) \equiv$$

$$\prod_{A_1, A_2, \ldots, A_n}(\sigma_F(\prod_{A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m}(E)))$$

关系代数表达式等价变换规则（续）

6. 选择与笛卡尔积的交换律

如果F中涉及的属性都是E_1中的属性，则

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

- 如果$F=F_1 \wedge F_2$，并且F_1只涉及E_1中的属性，F_2只涉及E_2中的属性，则由上面的等价变换规则1，4，6可推出：

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

- 若F_1只涉及E_1中的属性，F_2涉及E_1和E_2两者的属性，则仍有

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

它使部分选择在笛卡尔积前先做

关系代数表达式等价变换规则（续）

7. 选择与并的分配律

设$E=E_1 \cup E_2$，E_1，E_2有相同的属性名，则

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

8. 选择与差运算的分配律

若E_1与E_2有相同的属性名，则

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

9. 选择对自然连接的分配律

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

F只涉及E_1与E_2的公共属性

关系代数表达式等价变换规则（续）

10. 投影与笛卡尔积的分配律

设E_1和E_2是两个关系表达式，A_1，…，A_n是E_1的属性，B_1，…，B_m是E_2的属性，则

$$\prod_{A_1A_2\ldots A_nB_1B_2\ldots B_m}(E_1 \times E_2) \equiv \prod_{A_1A_2\ldots A_n}(E_1) \times \prod_{B_1B_2\ldots B_m}(E_2)$$

11. 投影与并的分配律

设E_1和E_2有相同的属性名，则

$$\prod_{A_1A_2\ldots A_n}(E_1 \cup E_2) \equiv \prod_{A_1A_2\ldots A_n}(E_1) \cup \prod_{A_1A_2\ldots A_n}(E_2)$$

9.3 代 数 优 化

- **9.3.1 关系代数表达式等价变换规则**
- **9.3.2 查询树的启发式优化**

9.3.2 查询树的启发式优化

- 典型的启发式规则：

1. 选择运算应尽可能先做

- 在优化策略中这是**最重要、最基本**的一条

2. 把投影运算和选择运算同时进行

- 如有若干投影和选择运算，并且它们都对同一个关系操作，则可以在扫描此关系的同时完成所有的这些运算以避免重复扫描关系

查询树的启发式优化（续）

3. 把投影同其前或其后的双目运算结合起来
   - 无需为了去掉某些字段而扫描一遍关系
4. 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
5. 找出公共子表达式
   - 如果这种重复出现的子表达式的结果不是很大的关系
   - 从外存中读入这个关系比计算该子表达式的时间少得多，则先计算一次公共子表达式并把结果写入中间文件是合算的
   - 当查询的是视图时，定义视图的表达式就是公共子表达式的情况

查询树的启发式优化（续）

- 遵循这些启发式规则，应用9.3.1的等价变换公式来优化关系表达式的算法。

  算法：关系表达式的优化

  输入：一个关系表达式的查询树

  输出：优化的查询树

  方法：

  - (1) 选择分解：利用等价变换规则4把形如 $\sigma_{F1\wedge F2\wedge\ldots\wedge Fn}(E)$变换为$\sigma_{F1}(\sigma_{F2}(\ldots(\sigma_{Fn}(E))\ldots))$
  - (2) 下推选择：对每一个选择，利用等价变换规则4～9尽可能把它移到树的叶端

规则4：选择的串接定律

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

规则4：　合并或分解选择运算
规则5-9：　选择运算与其他运算交换

查询树的启发式优化（续）

- **(3) 下推投影：**对每一个投影，利用等价变换规则3，5，10，11中的一般形式尽可能把它移向树的叶端。
- 注意：
  - 等价变换规则3使一些投影消失
  - 规则5把一个投影分裂为两个，其中一个有可能被移向树的叶端
  - 投影成的中间关系无索引

规则3：	合并或分解投影运算
规则5,10,11：	投影运算与其他运算交换

查询树的启发式优化（续）

- **(4) 选择投影串接合并：**利用等价变换规则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。使多个选择或投影能同时执行，或在一次扫描中全部完成
  - 可能违背“投影尽可能早做”原则，但效率更高

规则3：合并或分解投影运算
规则4：合并或分解选择运算
规则5：投影运算与选择运算交换

查询树的启发式优化（续）

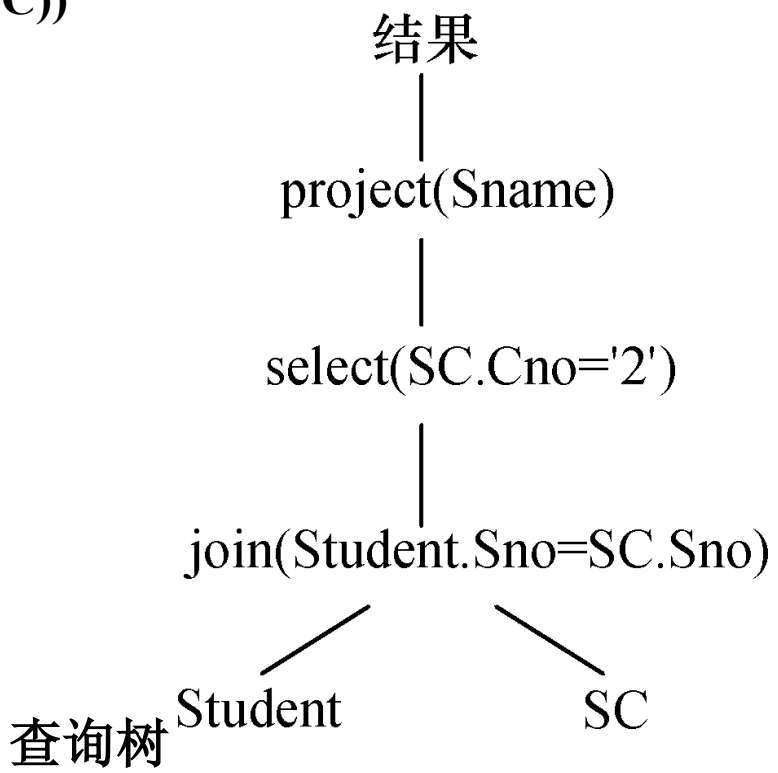
- **(5) 内节点分组：**把上述得到的语法树的内节点分组
  - 每一双目运算(×， ⋈ ，∪，－)和它所有的直接祖先为一组(这些直接祖先是(σ，∏运算)
  - 如果其后代直到叶子全是单目运算，则也将它们并入该组
  - 但当双目运算是笛卡尔积(×)，而且后面不是与它组成等值连接的选择时，则不能把选择与这个双目运算组成同一组，把这些单目运算单独分为一组

查询树的启发式优化（续）

［例9.4］ 下面给出［例9.3］中 SQL语句的代数优化示例。

(1) 把SQL语句转换成查询树，如下图所示

$$Q_1=\pi_{Sname}(\sigma_{Student.Sno = SC.Sno \wedge SC.Cno='2'}(Student \times SC))$$



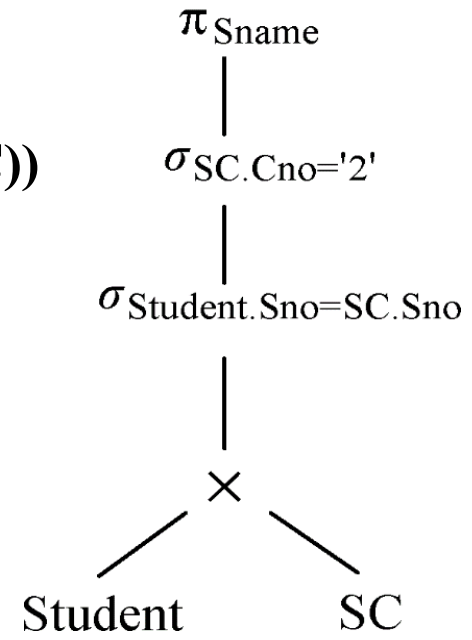
查询树

```
SELECT  Student.Sname
FROM  Student，SC
WHERE  Student.Sno=SC.Sno AND
    SC.Cno='2';
```

查询树的启发式优化（续）

- 为了使用关系代数表达式的优化法，假设内部表示是关系代a数语法树，则上面的查询树如下图所示

$$Q_1=\pi_{Sname}(\sigma_{Student.Sno = SC.Sno \wedge SC.Cno='2'}(Student \times SC))$$



关系代数语法树

```
SELECT  Student.Sname
FROM  Student，SC
WHERE  Student.Sno=SC.Sno AND
    SC.Cno='2';
```

m

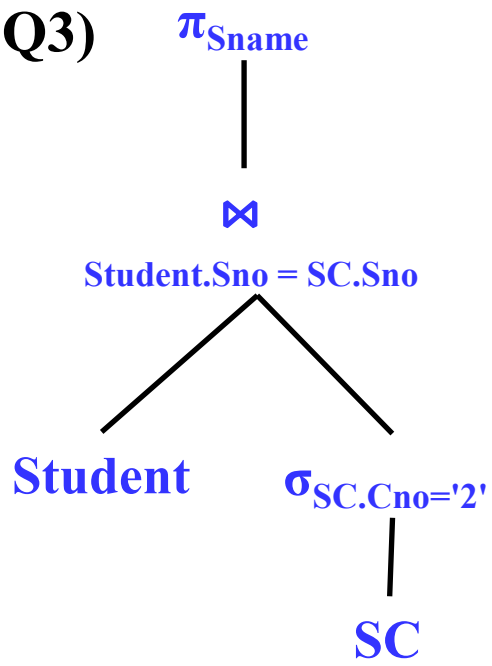
查询树的启发式优化（续）

(2) 对查询树进行优化

- 利用规则4、6把选择σ SC.Cno=‘2’移到叶端
- 启发式规则4：选择+笛卡尔积结合成为连接
- 查询树便转换成下图所示的优化的查询树(9.2.2中Q3)

$Q_1=\pi_{Sname}(\sigma_{Student.Sno = SC.Sno \wedge SC.Cno='2'}(Student \times SC))$

$Q_3=\pi_{Sname}(Student \bowtie \sigma_{SC.Cno='2'}(SC))$



优化后的查询树

9.1 关系数据库系统的查询处理

9.2 关系数据库系统的查询优化

9.3 代数优化

9.4 物理优化

9.5 小 结

9.4 物理优化

- 代数优化改变查询语句中操作的次序和组合，不涉及底层的存取路径
- 对于一个查询语句有许多存取方案，它们的执行效率不同， 仅仅进行代数优化是不够的
- 物理优化：选择高效合理的操作算法或存取路径，求得优化的查询计划
  - 选择
  - 连接

物理优化（续）

- 选择的方法：
  - 基于规则的启发式优化
    - 启发式规则是指那些在大多数情况下都适用，但不是在每种情况下都是适用的规则
  - 基于代价估算的优化
    - 优化器估算不同执行策略的代价，并选出具有最小代价的执行计划
  - 两者结合的优化方法
    - 常常先使用启发式规则，选取若干较优的候选方案，减少代价估算的工作量
    - 然后分别计算这些候选方案的执行代价，较快地选出最终的优化方案

9.4 物理优化

- **9.4.1 基于启发式规则的存取路径选择优化**
- **9.4.2 基于代价的优化**

9.4.1 基于启发式规则的存取路径选择优化

一、 选择操作的启发式规则

二、 连接操作的启发式规则

基于启发式规则的存取路径选择优化(续)

一、 选择操作的启发式规则：

- 对于小关系，使用全表顺序扫描（即使选择列上有索引）
- 对于大关系，启发式规则有：
  - （1）对于选择条件是“主码＝值”的查询
    - 查询结果最多是一个元组，可以选择主码索引
    - 一般的RDBMS会自动建立主码索引
  - （2）对于选择条件是“非主属性＝值”的查询，并且选择列上有索引
    - 要估算查询结果的元组数目
    - 如果比例较小(<10%)可以使用索引扫描方法
    - 否则还是使用全表顺序扫描

基于启发式规则的存取路径选择优化(续)

一、 选择操作的启发式规则：

- 对于大关系，启发式规则有：
  - （3）对于选择条件是属性上的非等值查询或者范围查询，并且选择列上有索引
    - 要估算查询结果的元组数目
    - 如果比例较小(<10%)可以使用索引扫描方法
    - 否则还是使用全表顺序扫描

基于启发式规则的存取路径选择优化(续)

一、 选择操作的启发式规则:

- 对于大关系，启发式规则有：
  - （4）对于用AND连接的合取选择条件
    - 如果有涉及这些属性的组合索引
      - 优先采用组合索引扫描方法
    - 如果某些属性上有一般的索引，可以用索引扫描方法
      - 通过分别查找满足每个条件的指针，求指针的交集
      - 通过索引查找满足部分条件的元组，然后在扫描这些元组时判断是否满足剩余条件［例9.1-C4］
    - 其他情况：使用全表顺序扫描
  - （5）对于用OR连接的析取选择条件，一般使用全表顺序扫描

基于启发式规则的存取路径选择优化(续)

二、 连接操作的启发式规则：

1. 如果2个表都已经按照连接属性排序
   - 选用排序-合并方法
2. 如果一个表在连接属性上有索引
   - 选用索引连接方法
3. 如果上面2个规则都不适用，其中一个表较小
   - 选用Hash join方法

基于启发式规则的存取路径选择优化(续)

二、 连接操作的启发式规则：

4. 可以选用嵌套循环方法，并选择其中较小的表，确切地讲是占用的块数(b)较少的表，作为外表(外循环的表) 。

- 设连接表R与S分别占用的块数为Br与Bs
- 连接操作使用的内存缓冲区块数为K
- 分配K-1块给外表
- 如果R为外表，则嵌套循环法存取的块数为Br+(Br/K-1)Bs
- 显然这样可以降低所需块数与I/O数，应该选块数小的表作为外表

9.4 物理优化（续）

- **9.4.1 基于启发式规则的存取路径选择优化**
- **9.4.2 基于代价的优化**

9.4.2 基于代价的优化

- 启发式规则优化是定性的选择，适合解释执行的系统
  - 解释执行的系统，优化开销包含在查询总开销之中
- 编译执行的系统中查询优化和查询执行是分开的
  - 可以采用精细复杂一些的基于代价的优化方法

基于代价的优化（续）

1.统计信息

2.代价估算示例

3.优化方法

基于代价的优化（续）

一、 统计信息

- 基于代价的优化方法要计算各种操作算法的执行代价，与数据库的状态密切相关
- 数据字典中存储的优化器需要的统计信息：
  - 1. 对每个基本表
    - 该表的元组总数(N)
    - 元组长度(l)
    - 占用的块数(B)
    - 占用的溢出块数(BO)

基于代价的优化（续）

- 2. 对基表的每个列
  - 该列不同值的个数(m)
  - 该列最大值
  - 该列最小值
  - 该列上是否已经建立了索引
  - 索引类型(B+树索引、Hash索引、聚集索引)
  - 计算选择率(f)
    - 如果不同值的分布是均匀的，f＝1/m，实际中做法
    - 如果不同值的分布不均匀，则每个值的选择率＝具有该值的元组数/N

基于代价的优化（续）

- 3. 对索引(如B+树索引)
  - 索引的层数(L)
  - 不同索引值的个数
  - 索引的选择基数S(有S个元组具有某个索引值)
  - 索引的叶结点数(Y)

基于代价的优化（续）

二、 代价估算示例

- **1. 全表扫描算法的代价估算公式**
  - **如果基本表大小为B块，全表扫描算法的代价 cost＝B**
  - **如果选择条件是码＝值，那么平均搜索代价 cost＝B/2**

基于代价的优化（续）

- 2. 索引扫描算法的代价估算公式
  - 如果选择条件是“码＝值”
    - 如［例9.1-C2］，则采用该表的主索引
    - 若为B+树，层数为L，需要存取B+树中从根结点到叶结点L块，再加上基本表中该元组所在的那一块，所以cost=L+1
  - 如果选择条件涉及非码属性
    - 如［例9.1-C3］，若为B+树索引，选择条件是相等比较，S是索引的选择基数(有S个元组满足条件)
    - 最坏的情况下，满足条件的元组可能会保存在不同的块上，此时，cost=L+S

［例9.1］Select * from student where <条件表达式>;

考虑<条件表达式>的几种情况

C1：无条件；
C2：Sno＝'200215121'；
C3：Sage>20；
C4：Sdept＝'CS' AND Sage>20；

基于代价的优化（续）

- 2. 索引扫描算法的代价估算公式
  - 如果比较条件是>，>=，<，<=操作
    - 假设有一半的元组满足条件，就要存取一半的叶结点
    - 通过索引访问一半的表存储块
    - **Cost = L + Y/2 + B/2**
    - 如果可以获得更准确的选择基数，可以进一步修正Y/2与B/2

基本表占用的块数(B)、索引的叶结点数(Y) 、索引的层数(L)

基于代价的优化（续）

- 3. 嵌套循环连接算法的代价估算公式
  - 嵌套循环连接算法的代价（ 9.4.1）

    cost＝Br+Bs/(K-1) Br
  - 如果需要把连接结果写回磁盘，

    cost＝Br+Bs/(K-1) Br + (Frs*Nr*Ns)/Mrs
    - Frs为连接选择性，表示连接结果元组数的比例
    - Mrs是存放连接结果的块因子，表示每块中可以存放的结果元组数目

基于代价的优化（续）

- 4. 排序-合并连接算法的代价估算公式
  - 如果连接表已经按照连接属性排好序，则 cost＝Br + Bs + (Frs*Nr*Ns)/Mrs。
  - 如果必须对文件排序
    - 需要在代价函数中加上排序的代价
    - 对于包含B个块的文件排序的代价大约是 $(2*B)+(2*B*\log_2 B)$

第九章 关系系统及其查询优化

9.5 小 结

- 查询处理是RDBMS的核心，查询优化技术是查询处理的关键技术
- 本章讲解的优化方法
  - 启发式代数优化
  - 基于规则的存取路径优化
  - 基于代价的优化
- 本章的目的：掌握查询优化方法的概念和技术

小　结（续）

- 比较复杂的查询，尤其是涉及连接和嵌套的查询
  - 不要把优化的任务全部放在RDBMS上
  - 应该找出RDBMS的优化规律，以写出适合RDBMS自动优化的SQL语句
- 对于RDBMS不能优化的查询需要重写查询语句，进行手工调整以优化性能