09 关系查询处理和查询优化

9.1 关系数据库系统的查询处理

9.1.1 查询处理步骤

RDBMS 查询处理阶段

1. *查询分析*：查询语句
    1. 对查询语句进行扫描、词法分析和语法分析
    2. 从查询语句中识别出语言符号
    3. 进行语法检查和语法分析
2. *查询检查*
    1. 根据数据字典对合法的查询语句进行语义检查
    2. 根据数据字典中的用户权限和完整性约束定义对用户的存取权限进行检查
    3. 检查通过后把 SQL 查询语句转换成等价的关系代数表达式
    4. RDBMS 一般都用查询书（语法分析树）来表示扩展的关系代数表达式
    5. 把数据库对象的外部名称转换为内部表示
3. *查询优化*
    1. 选择一个高效执行的查询处理策略
    2. 分类：代数优化、物理优化
    3. 优化方法选择依据：基于规则、代价、语义
4. *查询执行*
    1. 对依据优化器得到的执行策略生成查询计划
    2. 代码生成器生成执行查询计划的代码

9.1.2 实现查询操作的算法实例

选择操作的实现

```SQL
select * from Student where Sno = '200215121'
```

1. *简单的全表扫描方法*
    1. 对查询的基本表顺序扫描，逐一检查每个元组是否满足选择条件，把满足条件的元组作为结果输出
    2. 适合小表，不适合大表
2. *索引 / 散列扫描方法*
    1. 适合选择条件中的属性上有索引：B+树或 HASH 索引
    2. 通过索引先找到满足条件的元组主码或元组指针，再通过元组指针直接在查询的基本表中找到元组

连接操作的实现

- 连接操作是查询处理中最耗时的操作之一

```SQL
Select * From Student, SC
Where Student. Sno = SC. Sno
```

1. 嵌套循环方法

- 对外层 Student 的每一个元组 s，检索内层循环 SC 中的每一个元组 sc
- 检查这两个元组在连接属性 Sno 是否相等
- 如果满足连接条件，则串接后作为结果输出，直到外层循环表中的元组处理完为止
- *效率较低，尤其是对于大数据*

2. 排序 - 合并方法

- 适合连接的诸表已经排好序的情况
- *排序 - 合并连接方法的步骤*
  - 如果连接的表没有排好序，先对 Student 表和 SC 表按连接属性 Sno 排序
  - 取 Student 表中第一个 Sno，依次扫描 SC 表中具有相同 Sno 的元组
  - 当扫描到 Sno 不相同的第一个 SC 元组时，返回 Student 表，扫描它的下一个元组，再再扫描 SC 表中具有相同 Sno 的元组，把它们连接起来
  - 重复上述步骤直到 Student 表扫描完
- Student 表和 SC 表都只要扫描一遍
- 如果 2 个表原来无序，执行时间要加上对两个表的排序时间
- 对于 2 个大表，先排序后使用排序 - 合并方法执行连接，总的时间一般会大大减少

3. 索引连接方法

- 若 SC 表原来没有属性 Sno 上的索引，则建立该索引
- 对 Student 中每一个元组，由 Sno 值通过 SC 的索引查找相应的 SC 元组
- 把这些 SC 元组和 Student 元组连接起来
- 循环执行上述两步，直到 Student 表中的元组处理完为止

4. Hash Join 方法

- 选取两个表中的小表作为散列
- 将 R 和 S 的连接属性作为 Hash 码，用共同的 Hash 函数将两张表中的元组散列到同一个 Hash 文件中
- *划分阶段*：选择小表，将其元组按 Hash 函数分散到 Hash 表的桶中
- *试探阶段（连接阶段）*：对另一张表进行一遍处理，将其元组散列到适当的 Hash 桶中，并将元组与桶中所有来自小表并与之相匹配的元组连接起来

- *Hash Join 算法前提*：假设两个表中较小的表在第一阶段后可以完全放入内存的 Hash 桶中

9.2 关系数据库系统的查询优化

必要性：关系查询优化是影响 RDBMS 性能的关键因素
由于关系表达式的语义级别很高，使关系系统可以从关系表达式中分析查询语义，提供了执行查询优化的可能性

关系系统的查询优化

- 是关系数据库管理系统实现的关键技术又是关系系统的优点所在
- 减轻了用户选择存取路径的负担

非关系系统

- 用户使用过程化的语言表达查询要求，执行何种记录级的操作，以及操作的序列是由用户来决定的
- 用户必须了解存取路径，系统要提供用户选择存取路径的手段，查询效率由用户的存取策略决定
- 如果用户做了不当的选择，系统是无法对此加以改进的

查询优化的优点

- 用户不必考虑如何最好地表达查询以获得较好的效率
- 系统可以比用户程序的“优化”做得更好

RDBMS 通过某种**代价模型**计算出各种查询执行策略的执行代价，然后选取代价最小的执行方案
I/O 代价是最主要的
查询优化的总目标：选择有效策略，求得给定关系表达式的值，使得查询代价最小 (实际上是较小)

实际系统的查询优化步骤

1. 将查询转换成某种内部表示，通常是语法树
2. 根据一定的等价变换规则把语法树转换成标准 （优化）形式
3. 选择低层的操作算法
    1. 对于语法树中的每一个操作：计算各种执行算法的执行代价；选择代价小的执行算法
4. *生成查询计划（查询执行方案）*：查询计划是由一系列内部操作组成的

代数优化：即有选择和连接操作时，先做选择操作，这样参加连接的元组就可以大大减少
物理优化：对于 Student 和 SC 表的连接，利用 Student 表上的索引，采用 index join 代价也较小

实例

```SQL
Select Student. Sname
From Student, SC
Where Student. Sno = SC. Sno AND SC. Cno = '2'
```

9.3 代数优化

9.3.1 关系代数表达式等价变换规则

代数优化：基于关系代数等价变换规则的优化方法
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：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

常用的等价变换规则

1. 连接、笛卡尔积交换律
2. 连接、笛卡尔积结合律
3. 投影的串接定律
    1. $\Pi_{A_1,A_2,\ldots,A_n}(\Pi_{B_1,B_2,\ldots,B_m}(E))$等价于$\Pi_{A_1,A_2,\ldots,A_n}(E)$
    2. E 是关系代数表达式，A B 是属性名且 A 构成 B 的子集
4. 选择的串接定律
    1. E 是关系代数表达式，F_1、F_2 是选择条件
    2. 说明选择条件可以合并，这样一次就可检查全部条件
5. 选择与投影操作的交换律
6. 选择与笛卡尔积的交换律
7. 选择与并的分配律
8. 选择与差运算的分配律
9. 选择对自然连接的分配律
10. 投影与笛卡尔积的分配律
11. 投影与并的分配律

9.3.2 查询树的启发式优化

典型的启发式规则

- *重要、基本*：选择运算应尽可能先做，减小中间关系
- 在执行连接前对关系适当地预处理：建立索引、排序
- *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
- 把投影同其前或其后的双目运算结合起来，减少扫描关系的遍数
- 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，连接运算比笛卡尔积省很多时间

- *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并且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

① 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 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。

③ 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

④ 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

⑤ 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，π运算)。

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：要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

9.4.1 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，即使选择列上有索引
- *对于大关系，启发式规则有：*
  - 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*
  - 对于选择条件是 **非主属性 = 值** 的查询，并且选择列上有索引 / 选择条件是**属性上的非等值查询 / 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
    - 比例小于 10%可以使用索引扫描方法
    - 否则还是使用全表顺序扫描
  - 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，则可以用
      2. 通过索引先找到满足条件的元组主码或元组指针，再通过元组指针直接在查询的基本表中找到元组

      否则进行全表顺序扫描
  - 对于 **用OR连接的析取选择条件**，一般使用全表顺序扫描

连接操作的启发式规则

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