第三篇 系统篇

- 讨论数据库管理系统中查询处理和事务管理的基本概念和基础知识
  - 关系查询处理和查询优化
  - 数据库恢复
  - 并发控制

数据库系统概论
An Introduction to Database System

第九章 关系系统及其查询优化

中国人民大学信息学院 陈红

第九章　关系系统及其查询优化

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 查询执行

9.6 小 结

9.2 关系数据库系统的查询优化

- 查询优化在关系数据库系统中有着非常重要的地位
- 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

9.2 关系数据库系统的查询优化

- 9.2.1 查询优化的必要性
- 9.2.2 查询优化概述

9.2.1 查询优化的必要性

- 一个关系查询可以对应不同的执行方案，其效率可能相差非常大。

[例 3] 求选修了 2 号课程的学生姓名。用 SQL 表达：

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

- 假定学生 - 课程数据库中有 1000 个学生记录， 10000 个选课记录
- 其中选修 2 号课程的选课记录为 50 个

实例

- 系统可以用多种等价的关系代数表达式来完成这一查询

$Q1=\pi_{Sname}(\sigma_{Student.Sno=SC.Sno \wedge Sc.Cno='2'}(Student \times SC))$

$Q2=\pi_{Sname}(\sigma_{Sc.Cno='2'}(Student \bowtie SC))$

$Q3=\pi_{Sname}(Student \bowtie (\sigma_{Sc.Cno='2'}(SC)))$

实例（续）

- 一、第一种情况

$$Q1=\pi_{Sname}(\sigma_{Student.Sno=SC.Sno \wedge Sc.Cno='2'}(Student \times SC))$$

实例（续）

1. 计算广义笛卡尔积

- 算法：
  - 在内存中尽可能多地装入某个表（如 Student 表）的若干块，留出一块存放另一个表（如 SC 表）的元组。
  - 把 SC 中的每个元组和 Student 中每个元组连接，连接后的元组装满一块后就写到中间文件上
  - 从 SC 中读入一块和内存中的 Student 元组连接，直到 SC 表处理完。
  - 再读入若干块 Student 元组，读入一块 SC 元组
  - 重复上述处理过程，直到把 Student 表处理完

实例（续）

- 设一个块能装 10 个 Student 元组或 100 个 SC 元组，在内存中存放 5 块 Student 元组和 1 块 SC 元组，则读取总块数为

$$\frac{1000}{10} \quad \frac{1000}{10\times 5}\times\frac{10000}{100}$$

=100+20×100=2100 块

- 其中，读 Student 表 100 块。读 SC 表 20 遍，每遍 100 块。若每秒读写 20 块，则总计要花 105s
- 连接后的元组数为 $10^3\times10^4=10^7$。设每块能装 10 个元组，则写出这些块要用 $10^6/20=5\times10^4$s ≅ 13.89 小时

实例（续）

2. 作选择操作

- 依次读入连接后的元组，按照选择条件选取满足要求的记录
- 假定内存处理时间忽略。读取中间文件花费的时间（同写中间文件一样）需 5×10^4s
- 满足条件的元组假设仅 50 个，均可放在内存

实例（续）

3. 作投影操作

- 把第 2 步的结果在 Sname 上作投影输出，得到最终结果

❖第一种情况下执行查询的总时间

≈ $105+2\times5\times10^4\approx10^5$s ≈ 27.78 小时

- 所有内存处理时间均忽略不计

实例（续）

二、 第二种情况

$Q2=\pi_{Sname}(\sigma_{Sc.Cno='2'}(Student \bowtie C))$

- 1. 计算自然连接
  - 执行自然连接，读取 Student 和 SC 表的策略不变，总的读取块数仍为 2100 块花费 105 s
  - 自然连接的结果比第一种情况大大减少，为 10^4 个
  - 写出这些元组时间为 $10^4/10/20=50s$，为第一种情况的千分之一
- 2. 读取中间文件块，执行选择运算，花费时间也为 50s 。
- 3. 把第 2 步结果投影输出。
- 第二种情况总的执行时间≈ 105+50+50≈205s

实例（续）

三、 第三种情况

$Q3=\pi_{Sname}(\text{Student} \bowtie \sigma_{Sc.Cno='2'}(SC))$

- 1. 先对 SC 表作选择运算，只需读一遍 SC 表，存取 100 块花费时间为 5s，因为满足条件的元组仅 50 个，不必使用中间文件。
- 2. 读取 Student 表，把读入的 Student 元组和内存中的 SC 元组作连接。也只需读一遍 Student 表共 100 块，花费时间为 5s 。
- 3. 把连接结果投影输出
- 第三种情况总的执行时间≈ 5+5≈10s

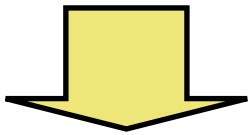
实例（续）

- 假如 SC 表的 Cno 字段上有索引
  - 第一步就不必读取所有的 SC 元组而只需读取 Cno='2' 的那些元组 (50 个)
  - 存取的索引块和 SC 中满足条件的数据块大约总共 3～4 块
- 若 Student 表在 Sno 上也有索引
  - 第二步也不必读取所有的 Student 元组
  - 因为满足条件的 SC 记录仅 50 个，涉及最多 50 个 Student 记录
  - 读取 Student 表的块数也可大大减少
- 总的存取时间将进一步减少到数秒

实例：小结

- 把代数表达式 *Q1* 变换为 *Q2* 、 *Q3*

$Q1=\pi_{Sname}(\sigma_{Student.Sno=SC.Sno \wedge Sc.Cno='2'}(Student \times SC))$



$Q2=\pi_{Sname}(\sigma_{Sc.Cno='2'}(Student \bowtie SC))$

$Q3=\pi_{Sname}(Student \bowtie (\sigma_{Sc.Cno='2'}(SC)))$

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，这是代数优化

实例：小结

- 在 Q3 中
  - SC 表的选择操作算法有全表扫描和索引扫描 2 种方法，经过初步估算，索引扫描方法较优
  - 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，这就是物理优化

9.2 关系数据库系统的查询优化

- 9.2.1 查询优化的必要性
- 9.2.2 查询优化概述

9.2.2 查询优化概述

- 关系系统的查询优化
  - 是 RDBMS 实现的关键技术又是关系系统的优点所在。它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

查询优化概述

- 查询优化的优点不仅在于用户不必考虑如何最好地表达查询以获得较好的效率，而且在于系统可以比用户程序的“优化”做得更好
  - (1) 优化器可以从数据字典中获取许多统计信息，而用户程序则难以获得这些信息
  - (2) 如果数据库的物理统计信息改变了，系统可以自动对查询重新优化以选择相适应的执行计划。在非关系系统中必须重写程序，而重写程序在实际应用中往往是不太可能的。

查询优化概述（续）

- (3) 优化器可以考虑数百种不同的执行计划，程序员一般只能考虑有限的几种可能性。
- (4) 优化器中包括了很多复杂的优化技术，这些优化技术往往只有最好的程序员才能掌握。系统的自动优化相当于使得所有人都拥有这些优化技术

查询优化概述（续）

- 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
  - 集中式数据库
    - 执行开销主要包括:
      - 磁盘存取块数 (I/O 代价)
      - 处理机时间 (CPU 代价)
      - 查询的内存开销
    - I/O 代价是最主要的
  - 分布式数据库
    - 总代价 =I/O 代价 +CPU 代价 + 内存代价 + 通信代价

查询优化概述（续）

- 查询优化的总目标：
  - 选择有效的策略
  - 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）