華中科技大學

数据库系统原理

第9章　关系查询优化

分级通关平台，QQ群

基础篇

第1章 绪论
第2章 关系数据库*3
第3章 标准语言SQL*3
第4章 数据库安全性
第5章 数据库完整性
实验1

设计与应用开发篇

第6章 关系数据理论*2
第7章 数据库设计
实验2
第8章 数据库编程

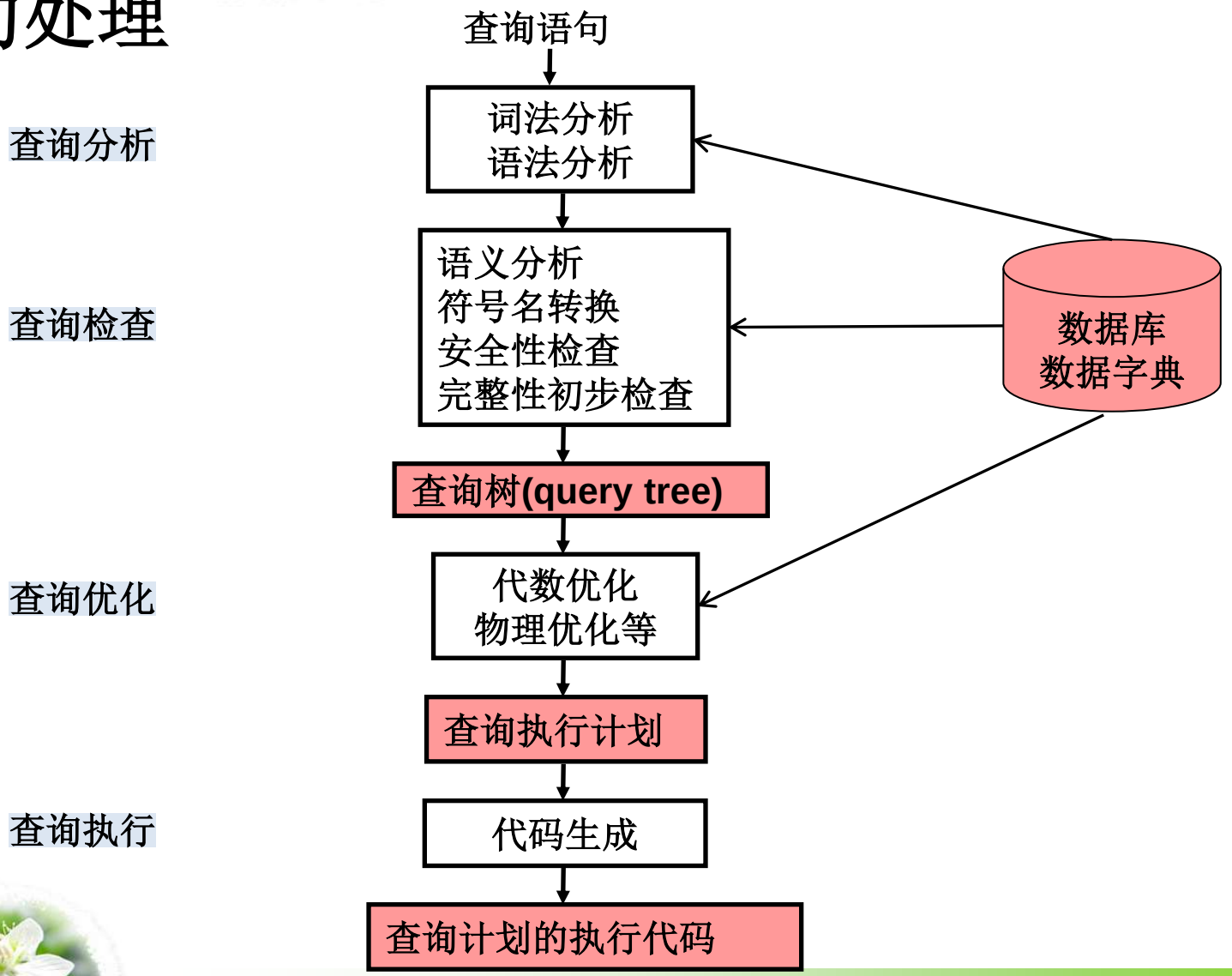
系统篇

第9章 *关系查询处理和优化
实验3
第10章 数据库恢复技术
第11章 并发控制
实验4

内容提要

- ✓ 关系数据库系统的查询处理
- ✓ 关系数据库系统的查询优化
- ✓ 代数优化
- ✓ 物理优化

查询的处理


查询语句
查询分析
词法分析
语法分析
查询检查
语义分析
符号名转换
安全性检查
完整性初步检查
数据库
数据字典
查询树(query tree)
查询优化
代数优化
物理优化等
查询执行计划
查询执行
代码生成
查询计划的执行代码

查询的处理

- 查询检查的任务
  - 合法权检查，视图转换
  - 安全性检查，完整性初步检查
    - 根据数据字典中有关的模式定义检查语句中的数据库对象，如关系名、属性名是否存在和有效
    - 用视图消解方法把对视图的操作转换成对基本表的操作
    - 用用户权限和完整性约束定义对用户的存取权限进行检查
    - 把SQL查询语句转换成内部表示，即等价的关系代数表达式
    - 用查询树，即语法分析树来表示扩展的关系代数表达式

查询的处理

- 查询优化：选择一个高效执行的查询处理策略
- [例9.1] SELECT * FROM Student

  WHERE <条件表达式>
  - C1：无条件
  - C2：Sno='201215121';
  - C3：Sage>20;
  - C4：Sdept='CS' AND Sage>20;
- 全表扫描算法(假设可以使用的内存为M块)：
  1. 按照物理次序读Student的M块到内存
  2. 检查内存的每个元组t，如果满足选择条件，则输出t
  3. 如果student还有其他块未被处理，重复①和②

查询的处理

- 查询优化：选择一个高效执行的查询处理策略
- [例9.1-C2] SELECT * FROM Student
  WHERE Sno='201215121'
  - 假设Sno上有索引(或Sno是散列码)
- 索引扫描算法
  - ✓ 使用索引(或散列)得到Sno为‘201215121’元组的指针
  - ✓ 通过元组指针在Student表中检索到该学生
    - ◆ [例9.1-C3] SELECT * FROM Student
      WHERE Sage>20
      - 假设Sage 上有B+树索引

查询的处理

- [例9.1-C4] SELECT * FROM Student

  WHERE Sdept='CS' AND Sage>20;

  - 假设Sdept和Sage上都有索引

✓ 算法一：

- 分别用Index Scan找到Sdept＝'CS'的元组指针和Sage>20的元组指针
- 求这两组指针的交集
- 到Student表中检索，得到计算机系年龄大于20的学生

✓ 算法二：

- 找到Sdept='CS'的元组指针，通过这些元组指针到Student表中检索
- 并对得到的元组检查另一些选择条件(如Sage>20)是否满足
- 把满足条件的元组作为结果输出。

查询优化

- 查询优化：选择一个高效执行的查询处理策略
- 查询优化分类
  - 代数优化/逻辑优化：指关系代数表达式的优化
  - 物理优化：指存取路径和底层操作算法的选择
- 查询优化的选择依据
  - 基于规则(rule based)
  - 基于代价(cost based)
  - 基于语义(semantic based)
- 代码生成器(code generator)生成执行查询计划的代码

查询优化—连接操作

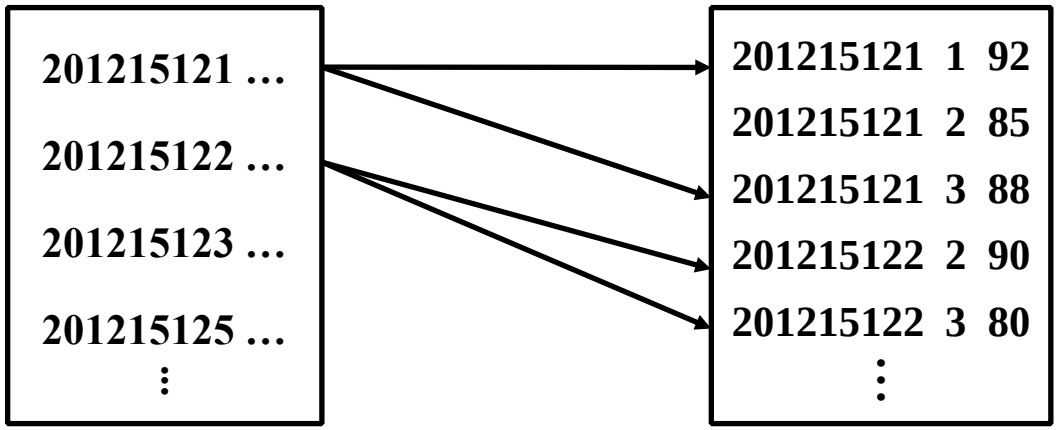
- 连接操作是查询处理中最耗时的操作之一
- [例9.2] SELECT * FROM Student, SC
  WHERE Student.Sno=SC.Sno;

1）嵌套循环算法(nested loop join)

2）排序-合并算法(sort-merge join 或merge join)

3）索引连接(index join)算法

4）Hash Join算法



查询优化—代数优化

- 关系查询优化是影响关系数据库管理系统性能的关键因素
  - 关系系统可以从关系表达式中分析查询语义，执行查询优化
- [例9.3] 求选修了2号课程的学生姓名。

```
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
  - 选修2号课程的选课记录为50个
- 可以用多种等价的关系代数表达式来完成这一查询

  - $Q_1=\Pi_{Sname}(\sigma_{Student.Sno=SC.Sno \wedge SC.Cno='2'}(Student \times SC))$
  - $Q_2=\Pi_{Sname}(\sigma_{SC.Cno='2'}(Student \bowtie SC))$
  - $Q_3=\Pi_{Sname}(Student \bowtie \sigma_{SC.Cno='2'}(SC))$
    - 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 =100+20×100=2100块

查询优化—代数优化

- 把代数表达式Q_1变换为Q_2、 Q_3

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

$$Q_2=\Pi_{Sname}(\sigma_{Sc.Cno='2'}(Student\bowtie SC))$$

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

- 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

代数优化

- 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
  - 关系代数表达式等价变换规则。11条交换律，结合律，分配律等。

- 典型的启发式规则

1）选择运算应尽可能先做

2）把投影运算和选择运算同时进行

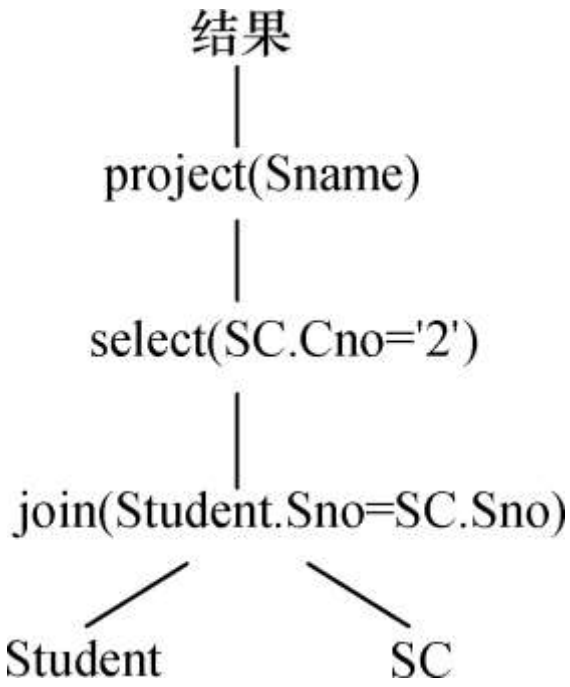
3）把投影同其前或其后的双目运算结合起来

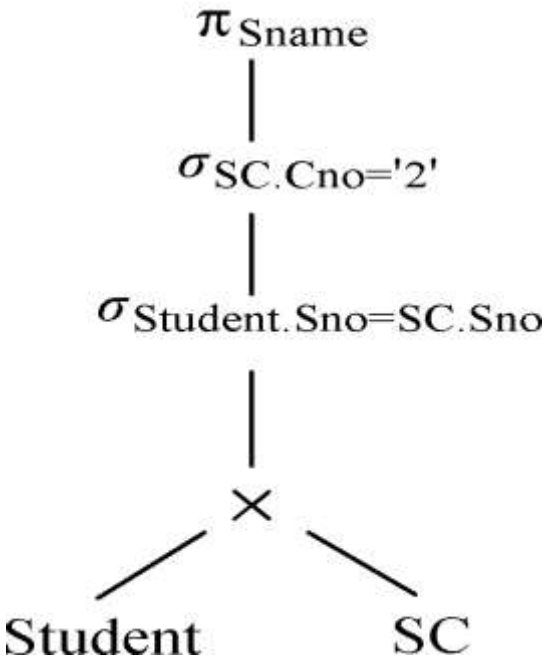
4）等值连接运算要比同样关系上的笛卡尔积省很多时间

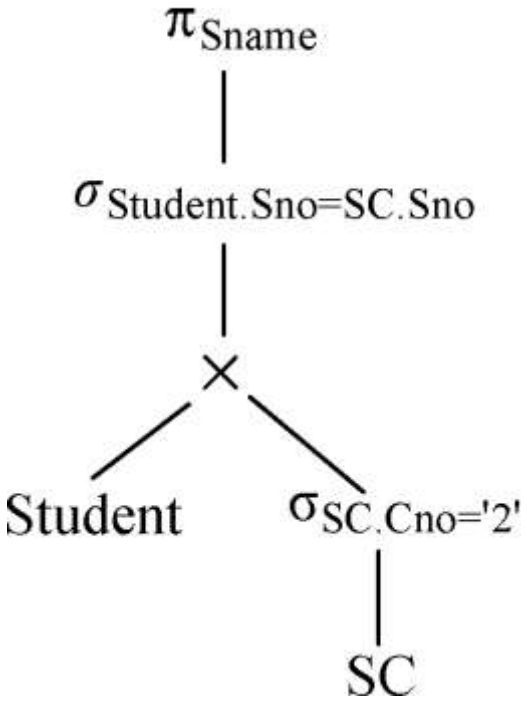
5）**找出公共子表达式：**构造中间结果文件，视图中构建公共表达式

✓ SELECT Student.Sno+':'+Student.Sname FROM Student, SC
WHERE Student.Sno=SC.Sno AND SC.Cno='2' ;

代数优化—查询树


结果
project(Sname)
select(SC.Cno='2')
join(Student.Sno=SC.Sno)
Student
SC


π Sname
σ SC.Cno='2'
σ Student.Sno=SC.Sno
×
Student
SC


π Sname
σ Student.Sno=SC.Sno
×
Student
σ SC.Cno='2'
SC

查询优化—物理优化

- 假如SC表的Cno字段上有索引
  - 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
  - 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

✓ SC表的选择操作算法有全表扫描或索引扫描，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，采用索引连接代价也较小，这就是物理优化。

物理优化

- 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，并选出最小代价的计划。
- 两者结合的优化方法
  - 先使用启发式规则，选取若干较优的候选方案
  - 然后分别计算这些候选方案的执行代价，选出最终的优化方案

1.选择操作的启发式规则

2.连接操作的启发式规则

物理优化

1.选择操作的启发式规则

- 对于小关系，即使选择列上有索引，也使用全表顺序扫描
- 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，选择主码索引

- 一般的关系数据库管理系统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

- 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

3）对于选择条件是属性上的非等值查询，并且选择列上有索引，同上

4）对于用AND连接的合取选择条件

- 用索引扫描方法，或使用全表顺序扫描

5）对于用OR连接的选择条件，一般使用全表顺序扫描

物理优化

2.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-合并算法

2）如果一个表在连接属性上有索引

- 选用索引连接算法

3）如果上面2个规则都不适用，其中一个表较小

- 选用Hash join算法

4）可以选用嵌套循环方法，并选择其中较小的表，作为外循环表。

◆ 有1000个学生记录，10000个选课记录，选课记录为50个

◆设一个块能装20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 =50+10×100=1050块

物理优化

- 基于代价估算的优化
  - 优化器估算不同执行策略的代价，并选出最小代价的计划。

1.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)

2）对基表的每个列

- 该列不同值的个数(m)，列最大值，最小值，列上是否已经建立了索引，哪种索引(B+树索引、Hash索引、聚集索引)，计算选择率(f)

3）对索引

- 索引的层数(L)，不同索引值的个数，索引的选择基数S，索引的叶结点数(Y)

物理优化

2.代价估算

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

2）索引扫描算法的代价估算公式

- ✓ 如果选择条件是“码＝值”，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，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)
  - 满足条件的元组可能会保存在不同的块上，所以(最坏的情况)cost=L+S
- ✓ 如果比较条件是＞，＞＝，＜，＜＝，假设有一半的元组满足条件，就要通过索引访问一半的表存储块，cost=L+Y/2+B/2

物理优化

2.代价估算

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，cost＝Br+BrBs/(K-1)
- ✓ 连接结果写回磁盘，cost＝Br+Br Bs/(K-1)+(Frs*Nr*Ns)/Mrs
- 其中Frs为连接选择性(join selectivity)，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

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，还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*log2B)

小结

- 查询处理过程
- 查询的代数优化
- 查询的物理优化
- 查询优化技术是查询处理的关键技术

- 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关系数据库管理系统上
  - 写出适合关系数据库管理系统自动优化的SQL语句
- 系统不能优化的查询需要重写查询语句，进行手工调整