查询处理（查询优化）

单 位：重庆大学计算机学院

引导问题

- C、Python、JAVA程序有没有优化呢？
- 优化带来什么好处？

主要学习目标

- 查询优化的目的
- 查询优化的步骤

关系代数如何表示查询

- SQL查询
  - 查找年龄大于20的学生姓名、年级
  - 查询“数据库”课程成绩大于80的学生姓名
- 如何用关系代数表示？

思考问题

- 一条SQL查询语句可以使用几种关系代数表示？
- 每一种表示的查询的查询代价是否相同？

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

查询优化概述

- 查询优化器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

(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

- 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

- 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)

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

   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。

[例1] 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

一 概述

- 系统可以用多种等价的关系代数表达式来完成这一查询

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

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

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

一 概述

- 一、第一种情况

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

1. 计算广义笛卡尔积

- 把Student和SC的每个元组连接起来的做法：
  - 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

一 概述

- 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

$$\frac{1000}{10}+\frac{1000}{10\times 5}\times\frac{10000}{100}$$ =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

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

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
  $\approx 105+2\times5\times10^4\approx10^5$s
- 所有内存处理时间均忽略不计

一 概述

- 二、 第二种情况

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

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

- 三、 第三种情况

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

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

一 概述

- 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

一 概述

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

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

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

- 把代数表达式Q_1变换为Q_2、 Q_3，
  - 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
- 在Q_3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

讨论1.查询优化涉及哪些方面，大致过程？

查询优化任务及过程

1)查询优化的基本过程?

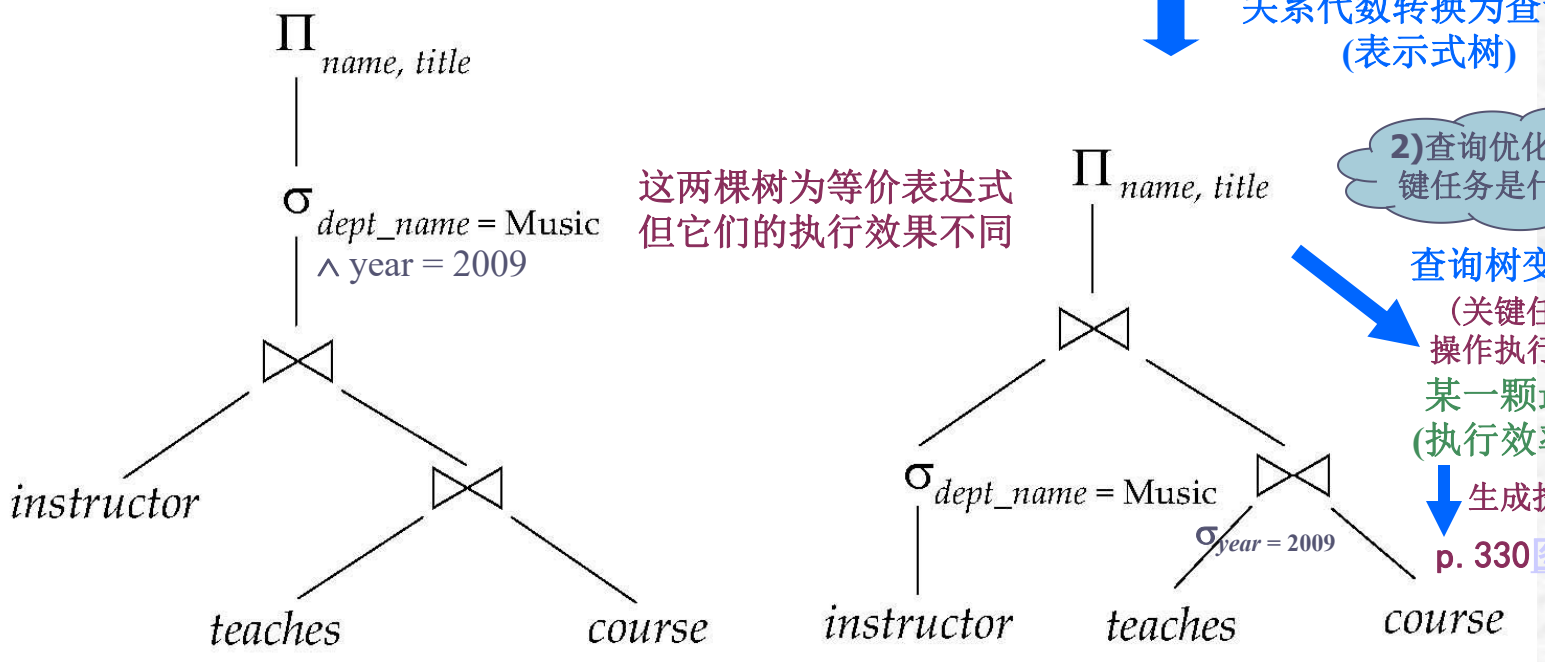
查询：2009年有课的Music系的所有教师名字以及每个教师所教授的课程名称

```
select name,title from instuctor,teaches,course
where instructor.ID=teaches.ID and teaches.course_id=course.course_id
      and instructor.dept_name='Music' and year=2009
```

$\Pi_{name,\ title}(\sigma_{dept\_name=\ \text{“Music”}\ \wedge year\ =\ 2009}$
$(instructor \bowtie (teaches \bowtie \Pi course\_id,\ title\ (course))))$

SQL转换为关系代数

关系代数转换为查询树(表示式树)



2)查询优化的关键任务是什么?

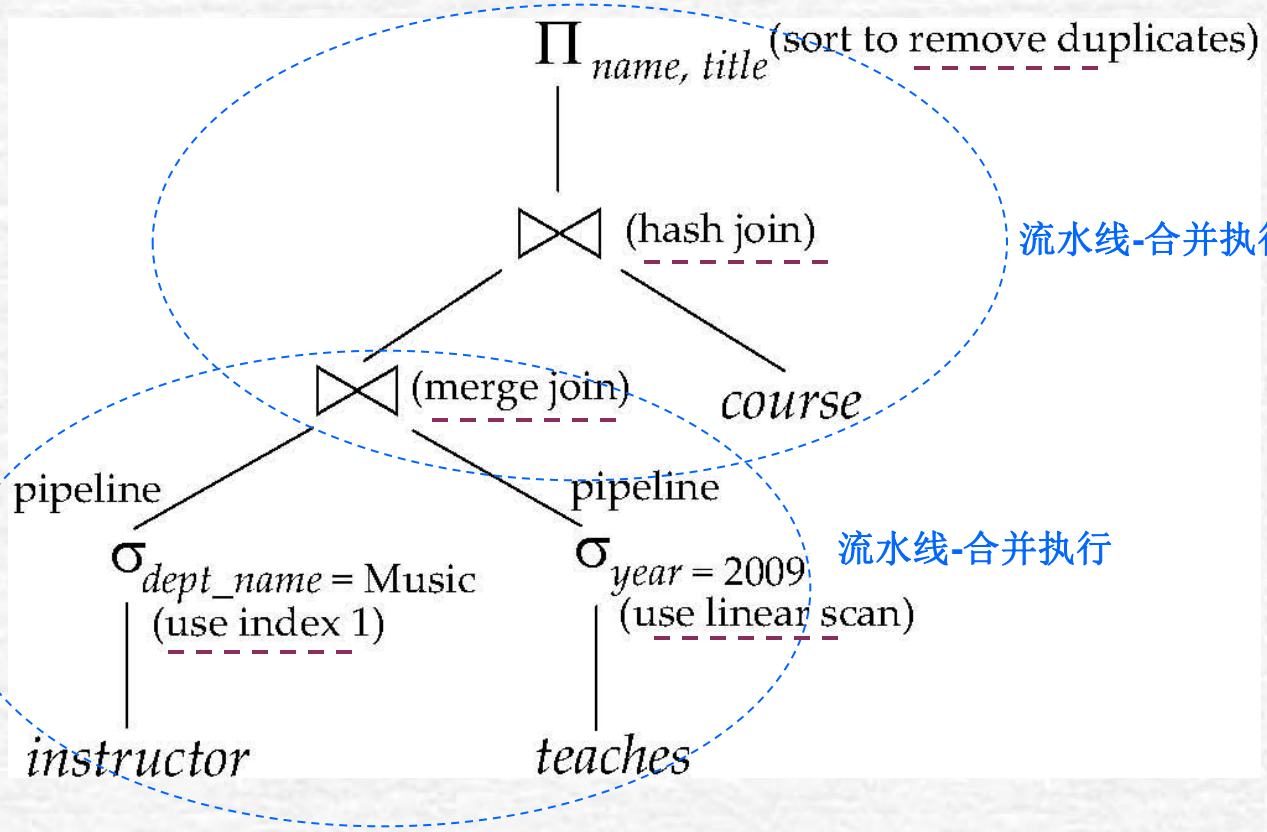
查询树变换
(关键任务)
操作执行次序
某一颗最优树
(执行效率最佳)

生成执行计划

p. 330图13-2

3) DBMS生成最终执行计划的内涵？

优化的查询树&执行计划



优化查询树 +标注（确定复合操作及确定各基本操作的执行算法）=执行计划

二 关系表达式的转换

讨论2.关系表达式进行等价转换的依据是什么？

1. 用于表达式转换的等价规则(略讲)

1)什么是等价规则，常用的有哪些?

1. 合取选择运算可分解为单个选择运算的序列。该变换称为 σ 的级联：

$$\sigma_{\theta_1 \wedge \theta_2}(E) = \sigma_{\theta_1}(\sigma_{\theta_2}(E))$$

2. 选择运算满足**交换律**(commutative)：

$$\sigma_{\theta_1}(\sigma_{\theta_2}(E)) = \sigma_{\theta_2}(\sigma_{\theta_1}(E))$$

3. 一系列投影运算中只有最后一个运算是必需的，其余的可省略。该转换也可称为 Π 的级联：

$$\Pi_{L_1}(\Pi_{L_2}(\cdots(\Pi_{L_n}(E))\cdots)) = \Pi_{L_1}(E)$$

4. 选择操作可与笛卡儿积以及 θ 连接相结合：

a. $\sigma_{\theta}(E_1 \times E_2) = E_1 \bowtie_{\theta} E_2$。

该表达式就是 θ 连接的定义。

b. $\sigma_{\theta 1}(E_1 \bowtie_{\theta_2} E_2) = E_1 \bowtie_{\theta_1 \wedge \theta_2} E_2$

5. θ 连接运算满足交换律：

$$E_1 \bowtie_{\theta} E_2 = E_2 \bowtie_{\theta} E_1$$

这些等价式说明什么？

等价规则（续1）

6. a. 自然连接运算满足**结合律**(associative)：

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

b. θ 连接具有以下方式的结合律：

$$(E_1 \bowtie_{\theta_1} E_2) \bowtie_{\theta_2 \wedge \theta_3} E_3 = E_1 \bowtie_{\theta_1 \wedge \theta_3} (E_2 \bowtie_{\theta_2} E_3)$$

$$(E_1 \bowtie_{\theta 1} E_2) \bowtie_{\theta 2 \wedge \theta 3} E_3 = E_1 \bowtie_{\theta 1 \wedge \theta 3} (E_2 \bowtie_{\theta 2} E_3)$$

7. 选择运算在下面两个条件下对 θ 连接运算具有分配律：

a. 当选择条件 θ_0 中的所有属性只涉及参与连接运算的表达式之一(比如 E_1)时，满足分配律：

$$\sigma_{\theta_0}(E_1 \bowtie_{\theta} E_2) = (\sigma_{\theta_0}(E_1)) \bowtie_{\theta} E_2$$

b. 当选择条件 θ_1 只涉及 E_1 的属性，选择条件 θ_2 只涉及 E_2 的属性时，满足分配律：

$$\sigma_{\theta_1 \wedge \theta_2}(E_1 \bowtie_{\theta} E_2) = (\sigma_{\theta_1}(E_1)) \bowtie_{\theta} (\sigma_{\theta_2}(E_2))$$

8. 投影运算在下面条件下对 θ 连接运算具有分配律：

a. 令 L_1、L_2 分别代表 E_1、E_2 的属性。假设连接条件 θ 只涉及 $L_1 \cup L_2$ 中的属性，则：

$$\Pi_{L_1 \cup L_2}(E_1 \bowtie_{\theta} E_2) = (\Pi_{L_1}(E_1)) \bowtie_{\theta} (\Pi_{L_2}(E_2))$$

b. 考虑连接 $E_1 \bowtie_{\theta} E_2$。令 L_1、L_2 分别代表 E_1、E_2 的属性集；令 L_3 是 E_1 中出现在连接条件 θ 中但不在 $L_1 \cup L_2$ 中的属性；令 L_4 是 E_2 中出现在连接条件 θ 中但不在 $L_1 \cup L_2$ 中的属性。那么：

$$\Pi_{L_1 \cup L_2}(E_1 \bowtie_{\theta} E_2) = \Pi_{L_1 \cup L_2}((\Pi_{L_1 \cup L_3}(E_1)) \bowtie_{\theta} (\Pi_{L_2 \cup L_4}(E_2)))$$

这些等价式说明什么？

等价规则（续2）

9. 集合的并与交满足交换律。

$$E_1 \cup E_2 = E_2 \cup E_1$$
$$E_1 \cap E_2 = E_2 \cap E_1$$

集合的差运算不满足交换律。

10. 集合的并与交满足结合律。

$$(E_1 \cup E_2) \cup E_3 = E_1 \cup (E_2 \cup E_3)$$
$$(E_1 \cap E_2) \cap E_3 = E_1 \cap (E_2 \cap E_3)$$

11. 选择运算对并、交、差运算具有分配律：

$$\sigma_P(E_1 - E_2) = \sigma_P(E_1) - \sigma_P(E_2)$$

类似地，上述等价规则将“-”替换成∪或∩时也成立。进一步有：

$$\sigma_P(E_1 - E_2) = \sigma_P(E_1) - E_2$$

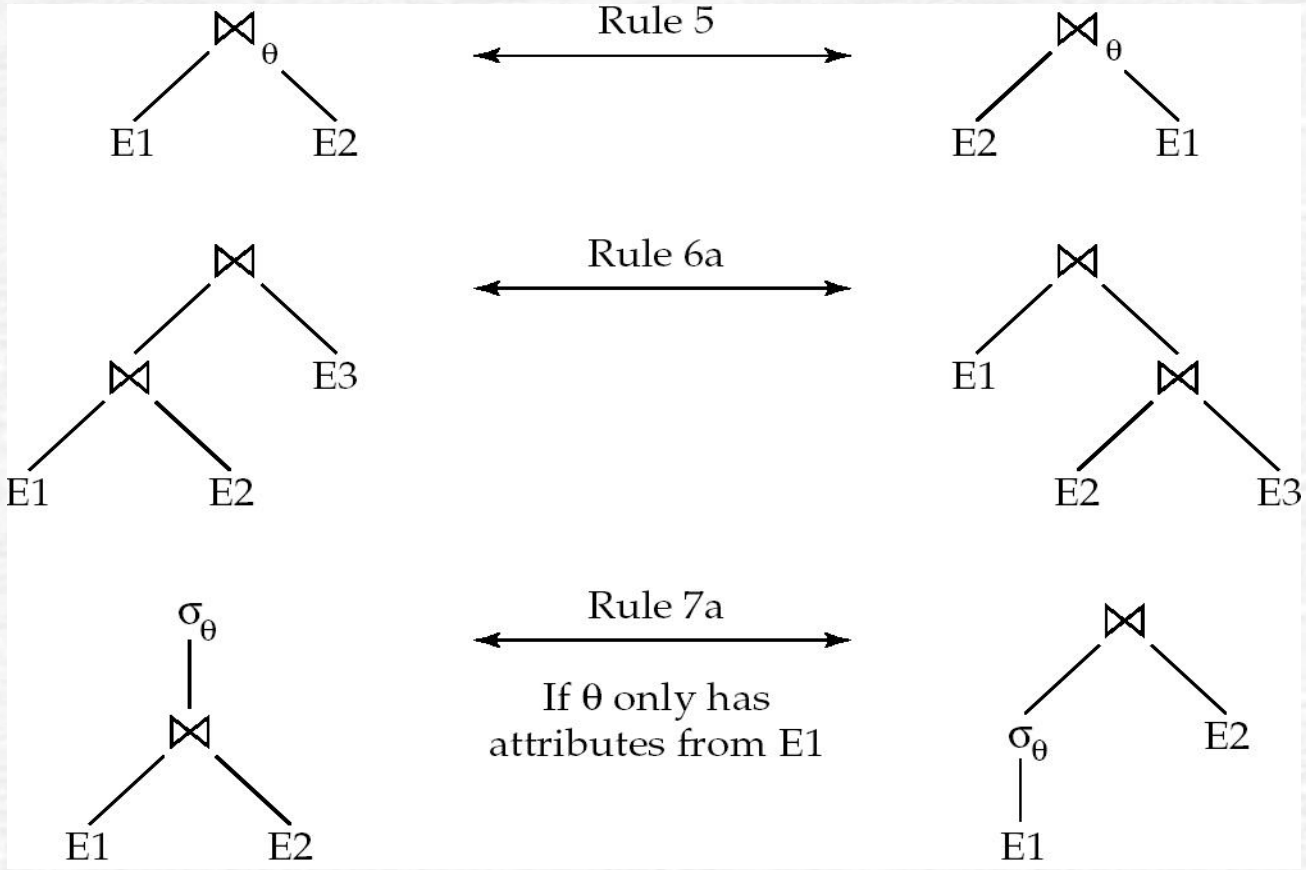
上述等价规则将“-”替换成∩时也成立，但将“-”替换成∪时不成立。

12. 投影运算对并运算具有分配律。

$$\Pi_L(E_1 \cup E_2) = (\Pi_L(E_1)) \cup (\Pi_L(E_2))$$

2. 等价规则的图示化

2)这些图(查询子树)说明什么？



θ连接的交换律！

自然连接的结合律！

选择对θ连接的分配律！

选择涉及单个关系时！

三 查询树的基本优化策略

1. “选择下移”优化策略

讨论3.如何利用等价规则进行关系表达式的转换？

1)“选择下移”有何优化效果？

查询：2009年有课的Music系的所有教师名字以及每个教师所教授的课程名称

$$\Pi_{name,\ title}(\sigma_{dept\_name=\ \text{“Music”}\ \wedge year\ =\ 2009}(instructor \bowtie (teaches \bowtie \Pi_{course\_id,\ title}(course))))$$

- 使用自然连接的结合律(**Rule 6a**):

$$\Pi_{name,\ title}(\sigma_{dept\_name=\ \text{“Music”}\ \wedge year\ =\ 2009}((instructor \bowtie teaches) \bowtie \Pi_{course\_id,\ title}(course)))$$

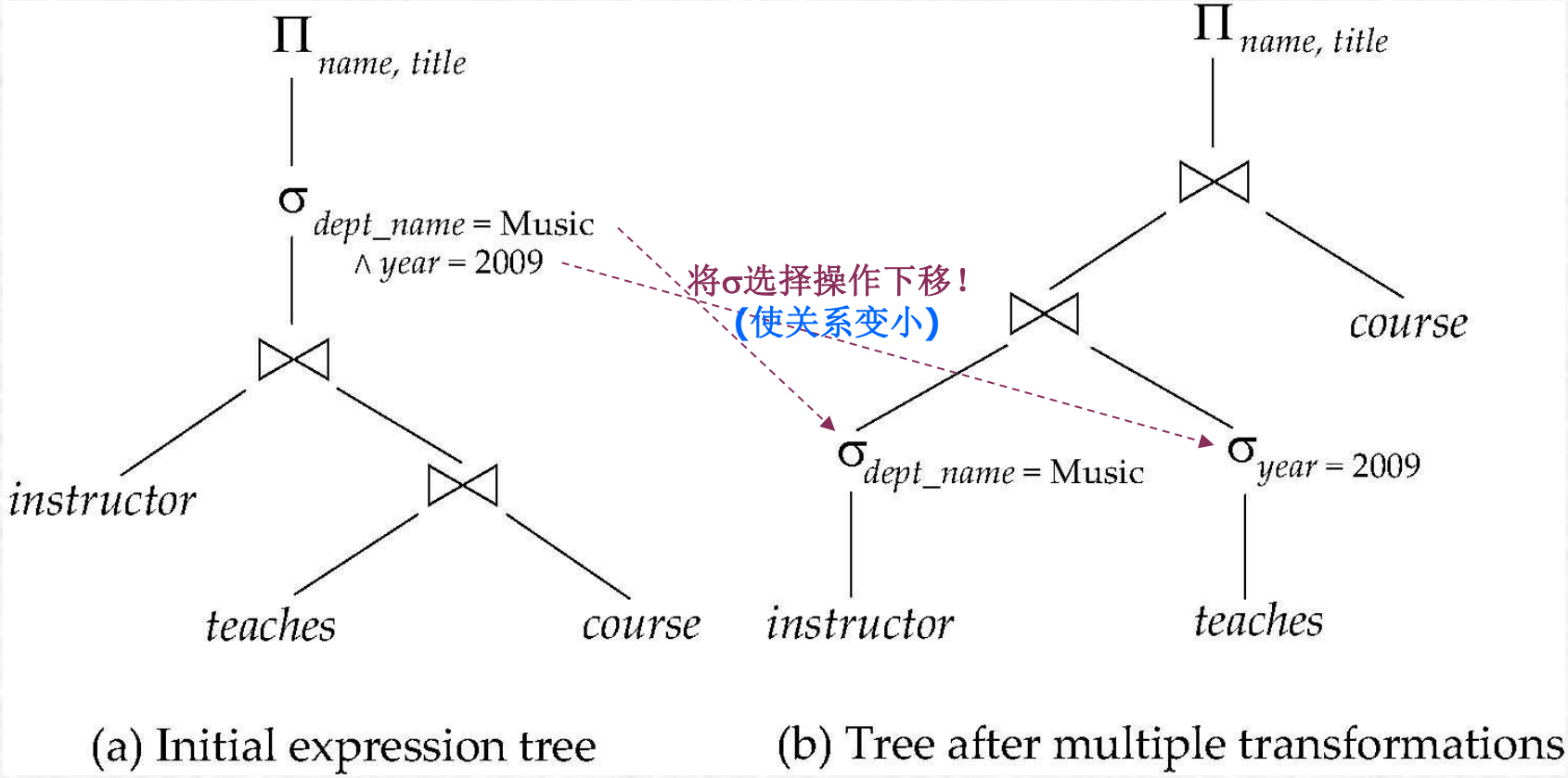
- 尽早执行选择

$$\Pi_{name,\ title}((\sigma_{dept\_name\ =\ \text{“Music”}}(instructor)) \bowtie \sigma_{year\ =\ 2009}(teaches))$$

- 查询树优化过程(图示法)

优化前、后的查询树有何不同？

“选择下移”查询树优化过程



2)"投影下移"有何优化效果?

2. "投影下移"优化策略

- 考虑: $\Pi_{name,\ title}((\sigma_{dept\_name=\ \text{“Music”}}(instructor) \bowtie teaches) \bowtie \Pi_{course\_id,\ title}(course))$

- When we compute

$\sigma_{dept\_name=\ \text{“Music”}}(instructor) \overset{id}{\bowtie} teaches$ 含8个属性的大关系!

we obtain a relation whose schema is:
(*ID, name, dept_name, salary, course_id, sec_id, semester, year*)

通过使用等价规则 8. a 及 8. b 先做投影运算，可以从模式中去除几个属性。

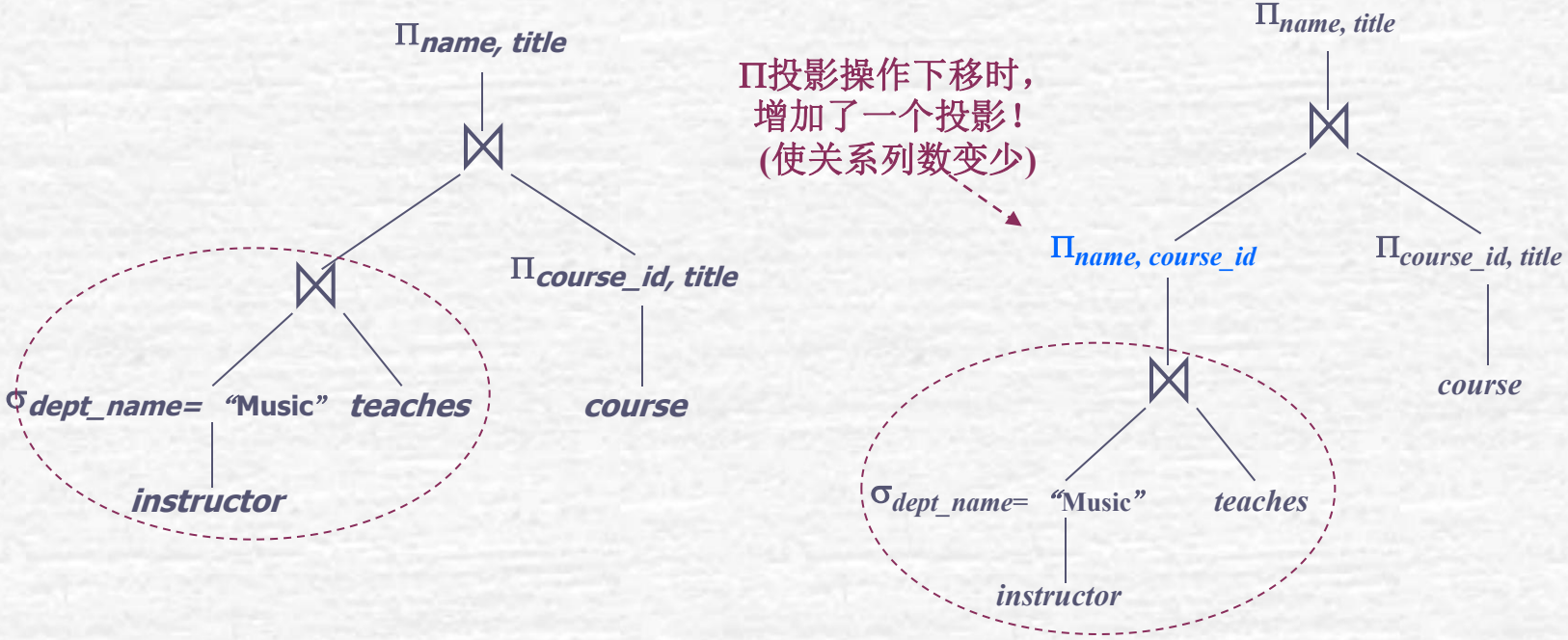
- $\Pi_{name,\ title}(\Pi_{name,\ course\_id}(\sigma_{dept\_name=\ \text{“Music”}}(instructor) \overset{id}{\bowtie} teaches) \overset{course\_id}{\bowtie} \Pi_{course\_id,\ title}(course))$

含2个属性的小关系!

- 通过去除不必要的属性，可以减少中间结果的列数，从而减少中间结果的大小。
- 查询树优化过程(图示法)

优化前、后的查询树有何不同？

“投影下移”查询树优化过程



3. “选择连接顺序”优化策略

3)"选择连接”次序有何优化效果？

- 考虑如下表达式:

$$\Pi_{name,\ title}((\sigma_{dept\_name=\ \text{“Music”}}(instructor) \bowtie teaches) \bowtie \Pi_{course\_id,\ title}(course))$$

- **能够首先计算**：$teaches \bowtie \Pi_{course\_id,\ title}(course)$, 然后在与下面表达式做连接

  $\sigma_{dept\_name=\ \text{“Music”}}(instructor)$ ？

  连接结果（记录数）更大

  是先做teaches ⋈ course好？还是instructor ⋈ teaches？

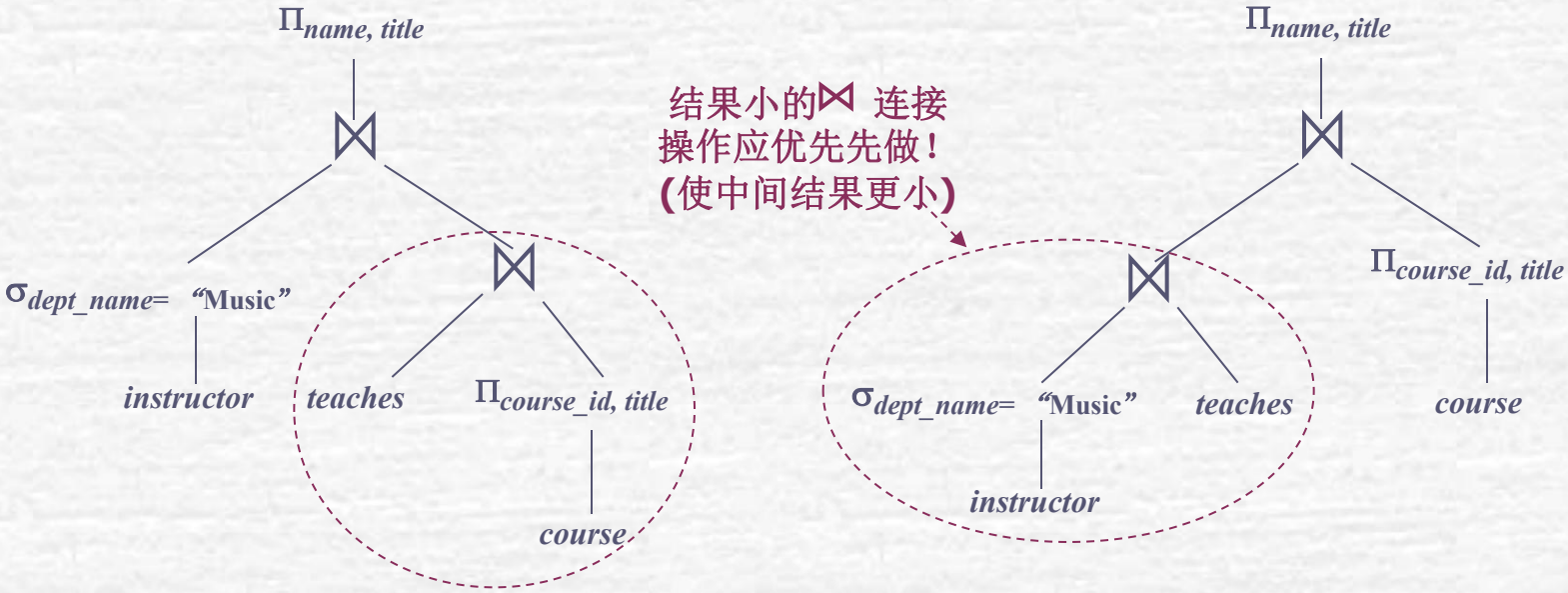
  但是第一个连接可能得到一个很大的结果.
- 大学老师中属于音乐学院的老师只有很少的一部分
- **因此最好先计算**
- $\sigma_{dept\_name=\ \text{“Music”}}(instructor) \bowtie teaches$

优化策略是：
小关系的连接应优先
(中间结果的元组数更少)

- **查询树优化过程(图示法)**

优化前、后的查询树有何不同？

“选择连接顺序”查询树优化过程



例子

- 查询：“张三”老师所在学院的预算和地址。

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

图 2-1 *instructor* 关系

dept_name	*building*	*budget*
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

图 2-5 *department* 关系

4. 执行计划

4)什么是“执行计划”，如何形成？

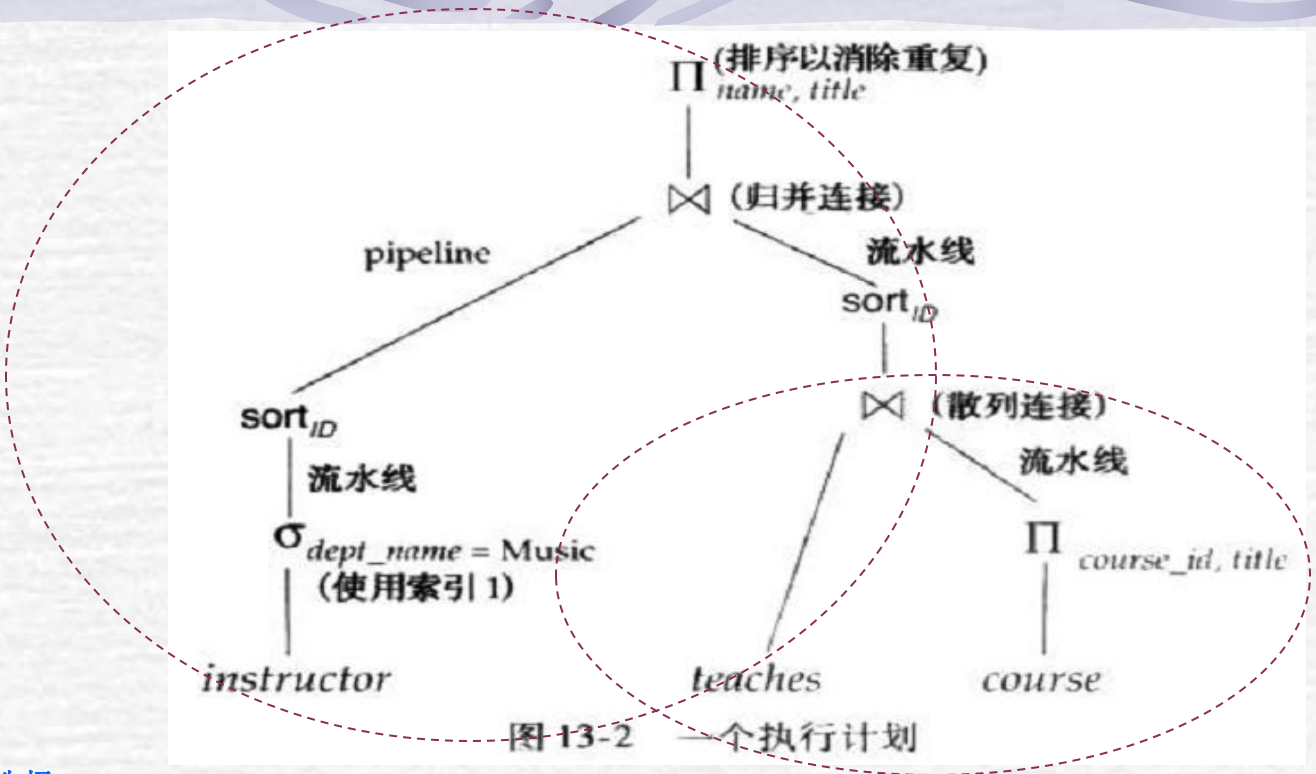


图 13-2 一个执行计划

算法选择

流水线(复合操作)

一个执行计划确切地定义了每个运算应使用的算法，以及运算之间的执行应该如何协调。图 13-2 表明了图 13-1b 的表达式的一个可行的执行计划。正如我们看到的，每个关系运算可以有几种不同的算法，从而可产生其他的执行计划。在图 13-2 中，连接运算中的一个选择了散列连接，另一个则在连接的 ID 属性上将关系排序后，选用归并连接。在凡被标记流水线的边上，生产者的输出直接流向消费者，而不会被写入磁盘。

给定一个关系代数表达式，查询优化器的任务是产生一个查询执行计划，该计划能获得与原关系表达式相同的结果，并且得到结果集的执行代价最小(或至少是不比最小执行代价大多少)。

四** 表达式结果集大小的估计(讲解)

1. 目录(统计)信息

返回

讨论4. 表达式转换，除了等价规则还需要考虑什么？

一个操作的代价依赖于它的输入的大小和其他统计信息。
比如，为计算(a ⋈ b ⋈ c)表达式，
我们需要有一些统计信息的估计，比如($b \bowtie c$)的大小。

2) 目录信息的主要用途？

数据库系统目录存储了有关数据库关系的下列统计信息：

- n_r，关系 r 的元组数。
- b_r，包含关系 r 中元组的磁盘块数。
- l_r，关系 r 中每个元组的字节数。
- f_r，关系 r 的块因子——一个磁盘块能容纳关系 r 中元组的个数。
- $V(A, r)$，关系 r 中属性 A 中出现的非重复值个数。该值与 $\Pi_A(r)$ 的大小相同。如果 A 是关系 r 的主码，则 $V(A, r)$ 等于 n_r。

目录信息在关系被修改时系统自动维护。不必精确，可以是统计得到（比如直方图），采用随机抽样法

1) 目录信息指什么，如何产生？

举一个直方图的例子来说，一个关系 *person* 的属性 *age* 的取值范围可分成 0 ~ 9，10 ~ 19，…，90 ~ 99(假设最大年龄值是99)。对每个区间，我们记录那些 *age* 值落在该区间的 *person* 元组个数，以及落入该区间的不同年龄取值的个数。如果没有这样的直方图信息，优化器将不得不假设属性值的分布是均匀的，即每个区间具有同样的计数值。

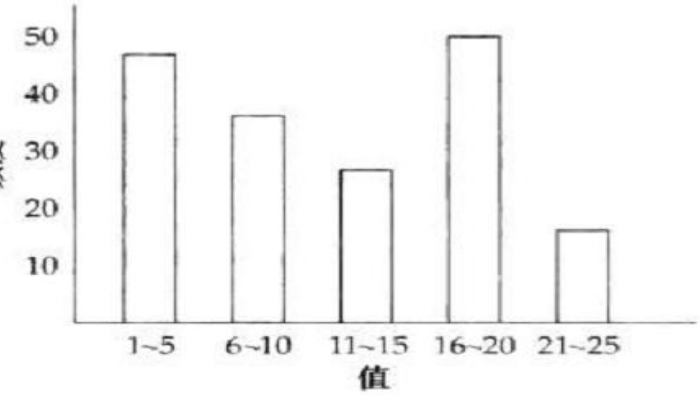


图 13-6 直方图示例

3）目录信息如何用于估计选择结果的大小？

查看参数说明

2. 选择运算结果大小的估计

A=a

- $\sigma_{A=a}(r)$：如果我们假设取值是均匀分布的（即，每个值以同样的概率出现），并假设关系 r 的一些记录的属性 A 中的取值为 a，则可估计选择结果有 $n_r/V(A,r)$ 个元组。这里选择操作中的值 a 在一些记录中出现的假设通常是成立的，而且代价估计总是默认这一假设。然而，假设每

A≤v

- $\sigma_{A\leqslant v}(r)$：考虑形如 $\sigma_{A\leqslant v}(r)$ 的选择操作。如果在进行代价估计时，用于比较操作的值（v）已知，则可做更精确的估计。属性 A 的最小值 $\min(A,r)$ 和最大值 $\max(A,r)$ 可存储到目录中。假设值是平均分布的，我们可以对满足条件 $A\leqslant v$ 的记录数进行下列估计：若 $v<\min(A,r)$，则为 0；若 $v\geqslant\max(A,r)$，则为 n_r；否则，为：

$$n_r \cdot \frac{v-\min(A,r)}{\max(A,r)-\min(A,r)}$$

4）目录信息如何用于估计连接结果的大小？

3. **连接运算结果大小的估计

分析连接连接属性的不同情况

估计自然连接的大小比估计选择或笛卡儿积的大小要更复杂一些。令 $r(R)$ 和 $s(S)$ 为两个关系。

- 若 $R \cap S = \emptyset$ ——两个关系没有共同的属性——则 $r \bowtie s$ 与 $r \times s$ 结果一样，我们可用估算笛卡儿积结果集大小的方法进行估计。 情形1
- 若 $R \cap S$ 是 R 的码，则我们可知 s 的一个元组至多与 r 的一个元组相连接。因此，$r \bowtie s$ 中的元组数不会超过 s 中元组的数目。$R \cap S$ 是 S 的码的情形同上面的情形相对称。若 $R \cap S$ 构成了 S 中参照 R 的外码，则 $r \bowtie s$ 中的元组数正好与 s 中元组数相等。 情形2 情形3
- 最难的情形是 $R \cap S$ 既不是 R 的码也不是 S 的码。在这种情况下，与进行选择运算的情况一样，我们假定每个值等概率出现。考虑 r 的元组 t，假定 $R \cap S = \{A\}$。我们估计元组 t 在 $r \bowtie s$ 中产生：

① $\frac{n_s}{V(A,s)}$ ② $\frac{n_r * n_s}{V(A,s)}$ 情形4 ③ $\frac{n_r * n_s}{V(A,r)}$ 情形5

① 个元组，因为该值就是关系 s 中在属性 A 上具有给定值的平均元组数目。考虑 r 中的所有元组，我们估计在 $r \bowtie s$ 中有：② 个元组。注意，如果将 r 与 s 的角色颠倒，那么我们估计在 $r \bowtie s$ 中有：

③ 个元组。在 $V(A,s) \neq V(A,r)$ 时，这两个估计是不同的。若发生这种情况，就可能有未参加连接的悬挂元组存在。因此这两个估计值中较小者可能比较准确。

估计示例(见下页)

**连接运算结果大小估计示例

举例说明对连接运算结果集大小的估计方法，考虑表达式

$$student \bowtie takes$$

假设有关这两个关系的目录信息如下：

- $n_{student} = 5000$
- $f_{student} = 50$，意味着 $b_{student} = 5000/50 = 100$
- $n_{takes} = 10\ 000$
- $f_{takes} = 25$，意味着 $b_{takes} = 10\ 000/25 = 400$
- $V(ID, takes) = 2500$，意味着只有一半的学生选课(这是不现实的，但是我们用它来表明在这个例子中我们的大小估计是正确的)，并且在平均情况下，选课的学生每人选 4 门课。

take 的属性 *ID* 是参照 *student* 的外码，而且 *take.ID* 上没有空值，因为 *ID* 是 *take* 主码中的一部分。因此 $student \bowtie takes$ 的大小恰好是 $n_{takes} = 10\ 000$。 属于情形3

下面我们在不使用外码信息情况下计算 $student \bowtie takes$ 的大小估计值。根据 $V(ID, takes) = 2500$，以及 $V(ID, student) = 5000$，我们得到两个估计值：$5000 * 10\ 000/2500 = 20\ 000$ 及 $5000 * 10\ 000/5000 = 10\ 000$，我们取较小者。在这里，估计值较小者与我们早先用外码信息计算所得结果相同。 属于情形4

属于情形5

查询优化-openGauss

查询重写

- 查询重写利用已有语句特征和关系代数运算来生成更高效的等价语句，在数据库优化器中扮演关键角色，尤其在复杂查询中，能够在性能上带来数量级的提升，可谓是“立竿见影”的“黑科技”。

查询优化-openGauss

- SQL是丰富多样的，应用非常灵活，不同的开发人员依据不同的经验，编写的SQL语句也是各式各样，SQL语句还可以通过工具自动生成。SQL是一种描述性语言，数据库的使用者只是描述了想要的结果，而不关心数据的具体获取方式。输入数据库的SQL语句很难做到以最优形式表示，往往隐含了冗余信息，这些信息可以被挖掘以生成更加高效的SQL语句。
- 查询重写就是把用户输入的 SQL 语句转换为更高效的等价SQL。
  查询重写遵循两个基本原则：

(1)等价性:原语句和重写后的语句输出结果相同。

(2)高效性:重写后的语句比原语句执行时间短，且资源使用更高效。

查询优化-openGauss

- openGauss几个关键的查询重写技术:常量表达式化简、子查询优化、 选择下推和等价推理、外连接消除、DISTINCT 消除、IN 谓词展开、视图展开等

https://mp.weixin.qq.com/s/ktJaRjBfOwV51ViM-xMhOA

查询优化-openGauss

1)常量表达式化简

- 常量表达式即用户输入的 SQL 语句中包含运算结果为常量的表达式，如算数表达式、逻辑运算表达式、函数表达式，查询重写可以对常量表达式预先计算以提升效率。

```
SELECT * FROM t1 WHERE c1=1+1;
SELECT * FROM t1 WHERE c1=2;
```

```
SELECT * FROM t1 WHERE 1=0 AND a=1;
SELECT * FROM t1 WHERE false;
```

```
SELECT * FROM t1 WHERE c1 = ADD(1, 1);
SELECT * FROM t1 WHERE c1=2;
```

查询优化-openGauss

2)子查询优化

- 由于子查询表示的结构更清晰，符合人们的阅读理解习惯，用户输入的SQL语句往往包含了大量的子查询。子查询有几种分类方法，根据子查询是否可以独立求解，分为相关子查询和非相关子查询。

① 相关子查询:

相关子查询是指子查询中有依赖父查询的条件，例如:

```
1  SELECT * FROM t1 WHERE EXISTS (SELECT t2.c1 FROM t2 WHERE t1.c1=t2.c1)
```

语句中子查询依赖父查询传入t1.c1的值。

② 非相关子查询:

非相关子查询是指子查询不依赖父查询，可以独立求解，例如:

```
1  SELECT * FROM t1 WHERE EXISTS (SELECT t2.c1 FROM t2);
```

查询优化-openGauss

- 相关子查询需要父查询执行出一条结果，然后驱动子查询运算，这种嵌套循环的方式执行效率较低。如果能把子查询提升为与父查询同级别，那么子查询中的表就能和父查询中的表直接做**Join(连接)**操作，由于**Join**操作可以有多种实现方法，优 化器就可以从多种实现方法中选择最优的一种，就有可能提高查询的执行效率

```
1  SELECT * FROM t1 WHERE t1.c1 IN (SELECT t2.c1 FROM t2);
2  SELECT * FROM t1 Semi Join t2 ON t1.c1 = t2.c1;
```

查询优化-openGauss

3)选择下推和等价推理

- 选择下推能够极大降低上层算子的计算量，从而达到优化的效果，如果选择条件存在等值操作，那么还可以借助等值操作的特性来实现等价推理，从而获得新的选择条件。

4)外连接消除

- 外连接和内连接的主要区别是对于不能产生连接结果的元组需要补充NULL值，如果SQL语句中有过滤条件符合空值拒绝的条件(即会将补充的NULL值再过滤掉)，则可以直接消除外连接。

5)DISTINCT 消除

- DISTINCT列上如果有主键约束，则此列不可能为空，且无重复值，因此可以不需要 DISTINCT操作，减少计算量。

查询优化-openGauss

6)IN 谓词展开

- 将IN 运算符改写成等值的过滤条件，便于借助索引减少计算量。

7)视图展开

- 视图从逻辑上可以简化书写SQL的难度，提高查询的易用性，而视图本身是虚拟的，因此在查询重写的过程中，需要展开视图。
- 可以将视图查询重写成子查询的形式，然后再对子查询做简化

随堂小测试

- 查询：“数据库”课程成绩大于80的学生姓名。
- 进行代数优化。

课程总结与作业安排

- 基本知识：
  - 查询优化的步骤
  - 代数优化的方法
- 扩展学习：
  - 查询优化器如何进行设计？
- 作业

  第13章习题：13.4，13.6 a)，13.7 b).
- 预习
  - 第16讲-事务管理（事务基本知识）（课前预习资料）
  - 简略介绍下一讲的学习内容

下一讲的学习内容

核心学习任务：事务概念

- 事务及其ACID特性
- 事务的状态
- 调度与可串行化调度
- 事务的可恢复性