第11讲 查询优化

教学内容

1. 查询优化概述
2. 逻辑查询优化
3. 物理查询优化
4. 代价估算

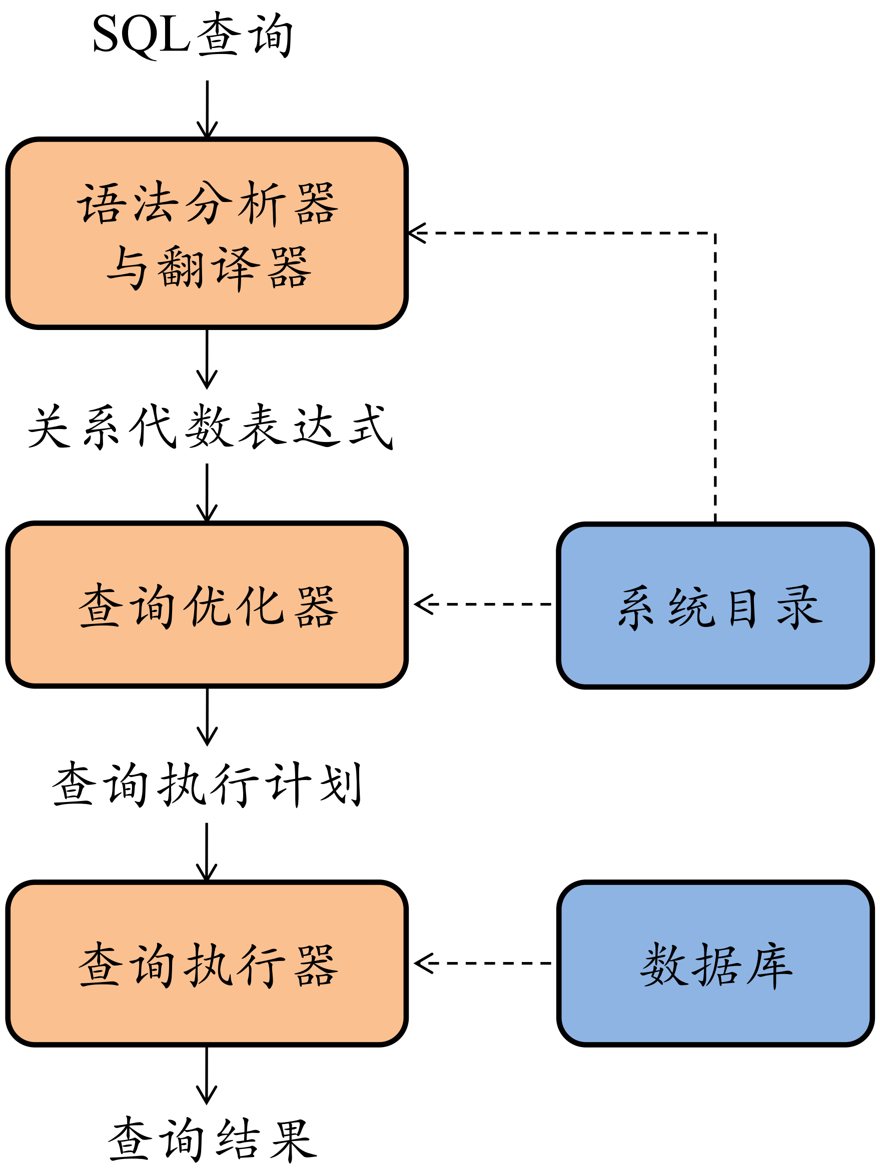
1. 查询优化概述

（1）查询处理的基本过程（回顾）

查询处理 (Query Processing) 是指从数据库中提取数据时涉及的一系列活动。
这些活动包括：将SQL查询语句翻译为能在文件系统的物理层上使用的表达式，为优化查询而进行各种转换，以及查询的实际执行。

基本过程包括：

1. 语法分析与翻译
2. 优化
3. 执行



1. 查询优化概述

（2）为什么要查询优化

关系数据库的执行效率问题

一个例子:

$$\Pi_{Sname}(\sigma_{SC.C\#=Course.C\# \wedge Student.S\#=SC.S\# \wedge Cname='DB'}(Student \times SC \times Course))$$

- Student: 10000个学生记录 (每年2500，四年)
- Course: 1000门课程记录
- SC: 10000*50=500000条选课记录 (10000学生，每人选50门课程)
- Student×SC×Course

  10000*50*10000*1000条记录=$5*10^{12}$条记录

1. 查询优化概述

（2）为什么要查询优化

关系代数操作执行次序对效率的影响

如下三条关系代数语句表示同样的检索需求，但哪一个更好呢?

$$\Pi_{S\#,Sname,Score}(\sigma_{Cname='DB'}((Student \bowtie SC) \bowtie Course))$$

10000　500000

500000　1000

500000

关系代数操作次序是否很重要呢?

$$\Pi_{S\#,Sname,Score}(Student \bowtie (SC \bowtie (\sigma_{Cname='DB'}Course)))$$

1000

500000　1

10000　2500(约)

2500(约)

$$\Pi_{S\#,Sname,Score}(\Pi_{S\#,Sname} Student \bowtie (SC \bowtie \Pi_{C\#}(\sigma_{Cname='DB'} Course)))$$

1. 查询优化概述

（3）什么是查询优化

查询优化：从众多查询策略中找出最有效的查询执行计划的处理过程，使数据库查询的执行时间最短

三个优化层面：

- 语义优化：利用模型的语义及完整性规则，优化查询
- 逻辑查询优化：利用语法结构，优化操作执行顺序
- 物理查询优化：存取路径和执行算法的选择与执行次序优化

教学内容

1. 查询优化概述
2. <u>逻辑查询优化</u>
3. 代价估算
4. 物理查询优化

2. 逻辑查询优化

（1）总体思路

select A1, …, An from R1, …, Rm where Cond ;

编译

$$\prod_{A1,...,An}(\sigma_{Cond}\ R1\times\ldots\times Rm)$$

语法优化/逻辑优化

$$\prod_{A1,...,An}(\sigma_{Cond}((\prod_{a1}(\sigma_{Cond1}R1))\times\ldots\times(\prod_{am}(\sigma_{Condm}Rm))))$$

逻辑查询计划——初步可按此执行

语法等价性，即形式上等价

逻辑优化/语法优化

基本思想：改变关系代数的操作次序，尽可能早做选择和投影运算

关系代数的五种基本操作中哪两个能够交换次序呢?

次序改变前后两个表达式的等价性问题

关系代数表达式的等价变换定理及其证明(略)

关系代数表达式的优化算法－逻辑查询计划形成

2. 逻辑查询优化

（2）关系代数操作次序交换的等价规则

关系代数表达式的等价性：设E_1，E_2是两个关系操作表达式，若E_1，E_2在每一个有效的数据库实例上都产生相同的元组集，则说E_1，E_2是等价的，记为$E_1 \equiv E_2$。

即，对于同样数据，输出一定相同。

$$\Pi_{S\#,Sname,Score}(\sigma_{Cname='DB'}((Student \bowtie SC) \bowtie Course))$$

$$\Pi_{S\#,Sname,Score}(Student \bowtie (SC \bowtie (\sigma_{Cname='DB'} Course)))$$

关系代数的基本操作

并　差　积　投影　选择

得到等价表达式的可能方式：

1. 某些二元操作的左右表达式交换顺序
2. 表达式内某些操作交换执行顺序
3. 添加或减少部分操作
4. 改写（拆分、合并、替换）部分操作

需要具体验证是否等价。参照L1-L10定理

2. 逻辑查询优化

（2）关系代数操作次序交换的等价规则

定理L1：连接与连接，积与积的交换律

设E_1, E_2是关系代数表达式，F是E_1, E_2中属性的附加限制条件，则有：

（1）$E_1 \bowtie_F E_2 = E_2 \bowtie_F E_1$　　(Θ-连接)

（2）$E_1 \bowtie E_2 = E_2 \bowtie E_1$　　(自然连接)

（3）$E_1 \times E_2 = E_2 \times E_1$　　(笛卡尔积)

注：

1. 并运算、交运算也有这种交换律，**差运算没有!**
2. 事实上交换后输出属性顺序不同。这里忽略该差异。

并　交　积　连接

2. 逻辑查询优化

（2）关系代数操作次序交换的等价规则

定理L2：连接与连接、积和积的结合律

设E_1, E_2, E_3是关系代数表达式， F_1, F_2是条件，则有：

（1） $(E_1 \bowtie_{F1} E_2) \bowtie_{F2} E_3 = E_1 \bowtie_{F1} (E_2 \bowtie_{F2} E_3)$

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

（3） $(E_1 \times E_2) \times E_3 = E_2 \times (E_1 \times E_2)$

注：并运算、交运算也有这种集合律，差运算没有。

select * from R1, R2, R3, R4 where ...

怎样确定FROM子句后的乘积次序?

2. 逻辑查询优化

（2）关系代数操作次序交换的等价规则

定理L3：投影串接律

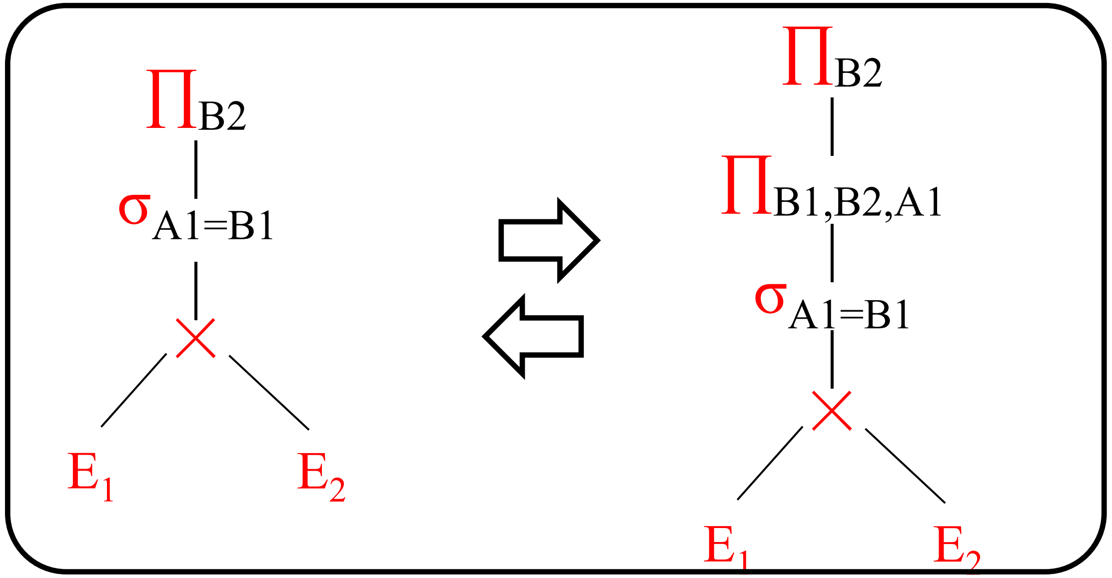
设属性集$\{A_1,\ldots,A_n\}\subseteq\{B_1,\ldots,B_m\}$，E是表达式，则有：

$$\prod_{A1,\ldots,An}(\prod_{B1,\ldots,Bm}E)\equiv\prod_{A1,\ldots,An}E$$

此定理可双向使用

多遍扫描变为一遍扫描

属性扩展便于投影操作的移动



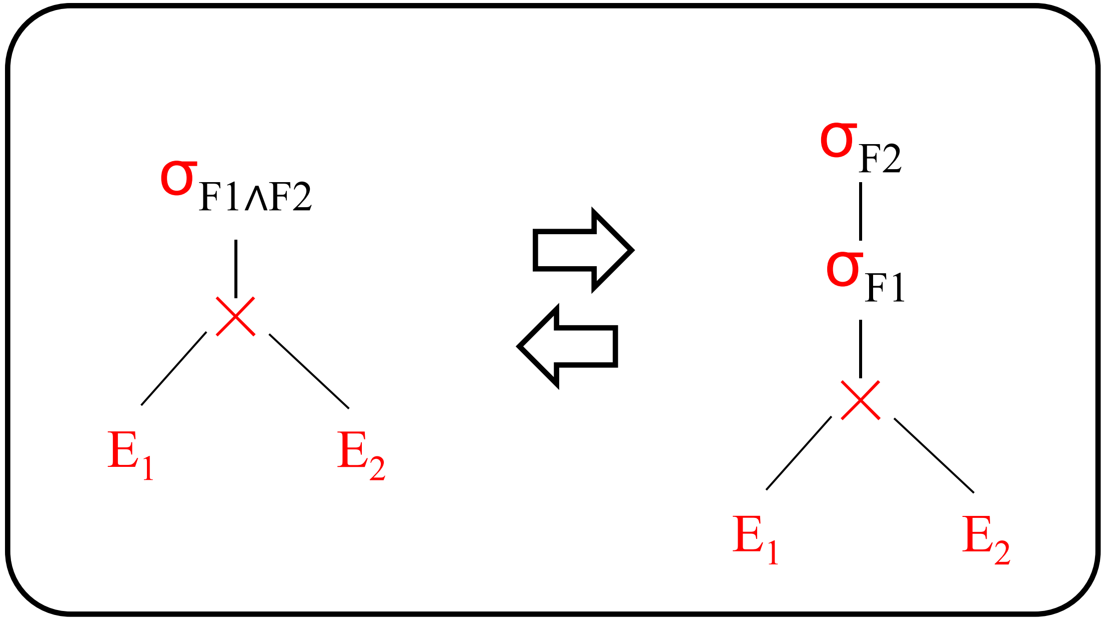
2. 逻辑查询优化

（2）关系代数操作次序交换的等价规则

定理L4：选择串接律

设E是关系代数表达式，F_1, F_2是条件，则有：

$$\sigma_{F1}(\sigma_{F2}E) \equiv \sigma_{F1 \wedge F2}E$$



此定理可双向使用

多遍扫描变为一遍扫描

分解复杂操作便于选择操作的移动

(2) 关系代数操作次序交换的等价规则

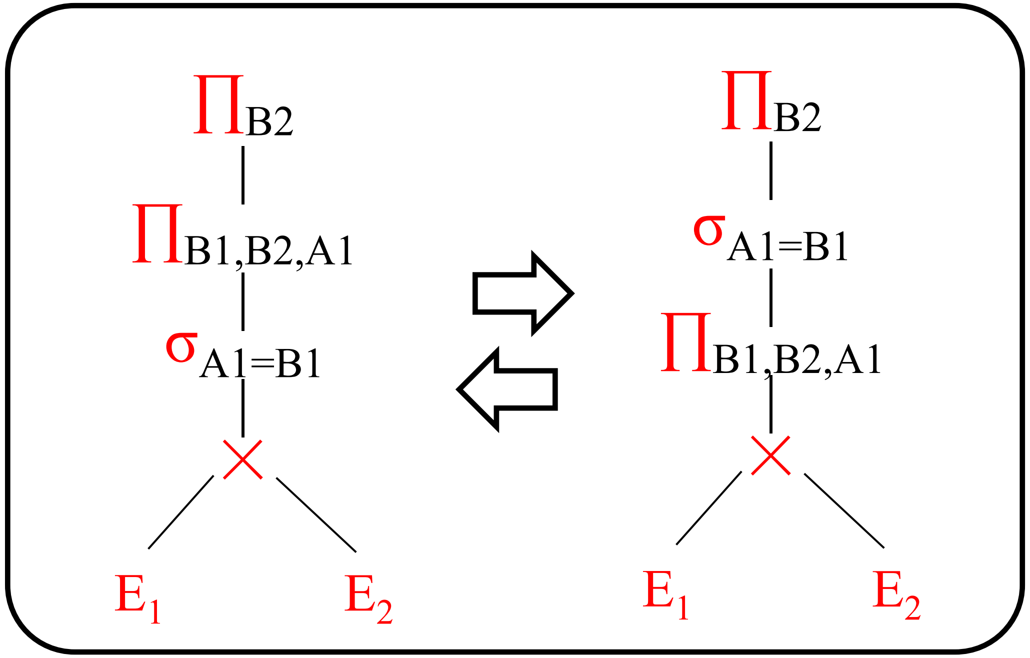
定理L5：选择和投影交换律

设条件F只涉及属性$\{A_1,\ldots, A_n\}$，E是关系表达式，则有：

$$\Pi_{A1,\ldots,An}(\sigma_F E) \equiv \sigma_F(\Pi_{A1,\ldots,An} E)$$

更一般地，若F还涉及不属于$\{A_1,..,A_n\}$的属性$\{B_1,..,B_m\}$，则有

$$\Pi_{A1,\ldots,An}(\sigma_F E) \equiv \Pi_{A1,\ldots,An}(\sigma_F(\Pi_{A1,\ldots,An,B1,\ldots,Bm} E))$$



2. 逻辑查询优化

（2）关系代数操作次序交换的等价规则

定理L6：选择和积的交换律

设E_1, E_2是关系代数表达式

- 若条件F只涉及E_1中的属性，则有：

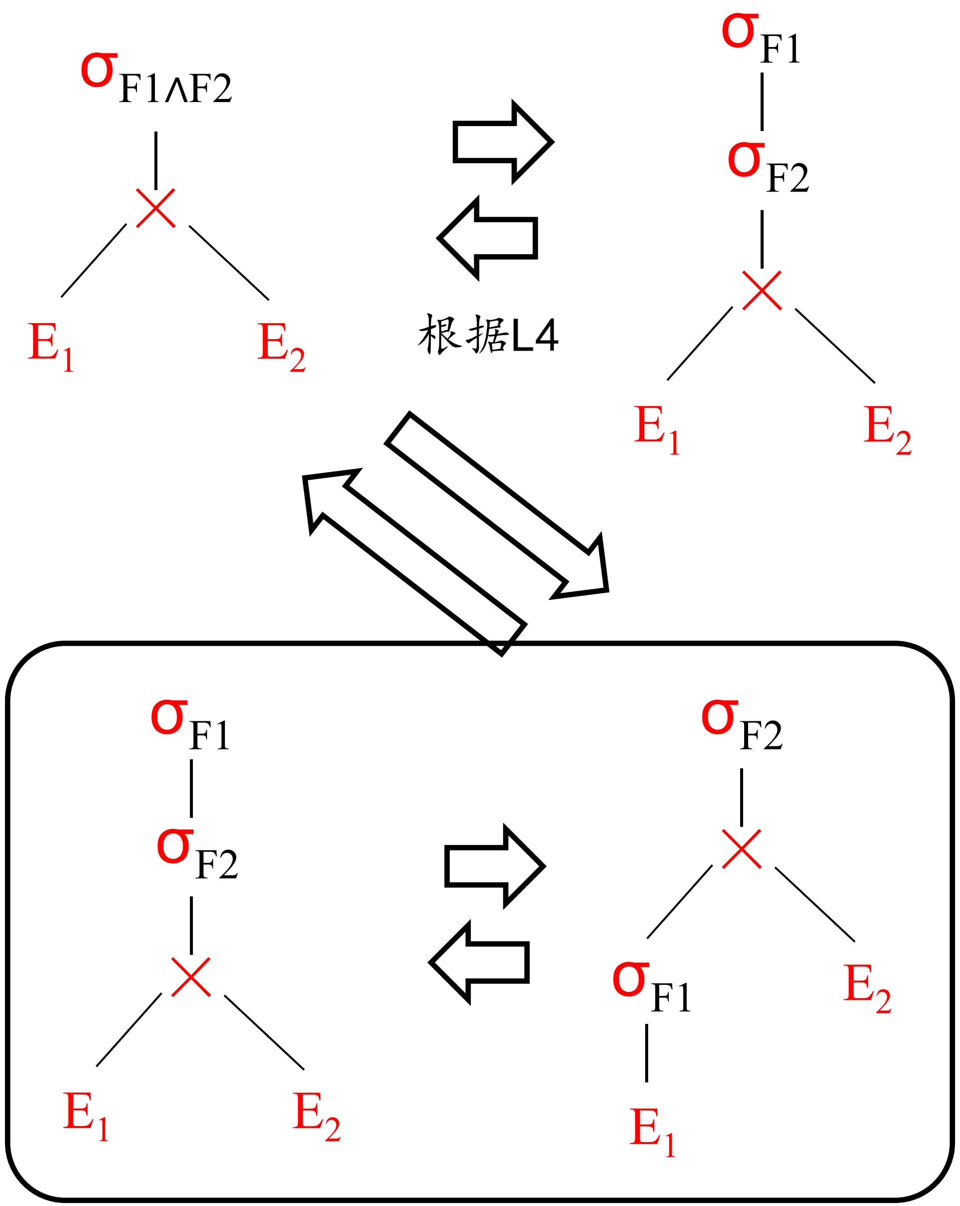
$$\sigma_F E_1 \times E_2 \equiv \sigma_F (E_1) \times E_2$$

- 若$F = F_1 \wedge F_2$，F_1, F_2分别只涉及E_1, E_2中属性，有：

$$\sigma_F E_1 \times E_2 \equiv \sigma_{F1} (E_1) \times \sigma_{F2} (E_2)$$

- 若$F = F_1 \wedge F_2$，F_1只涉及E_1中属性，而F_2涉及E_1和E_2中属性，则有：

$$\sigma_F E_1 \times E_2 \equiv \sigma_{F2} (\sigma_{F1} (E_1) \times E_2)$$



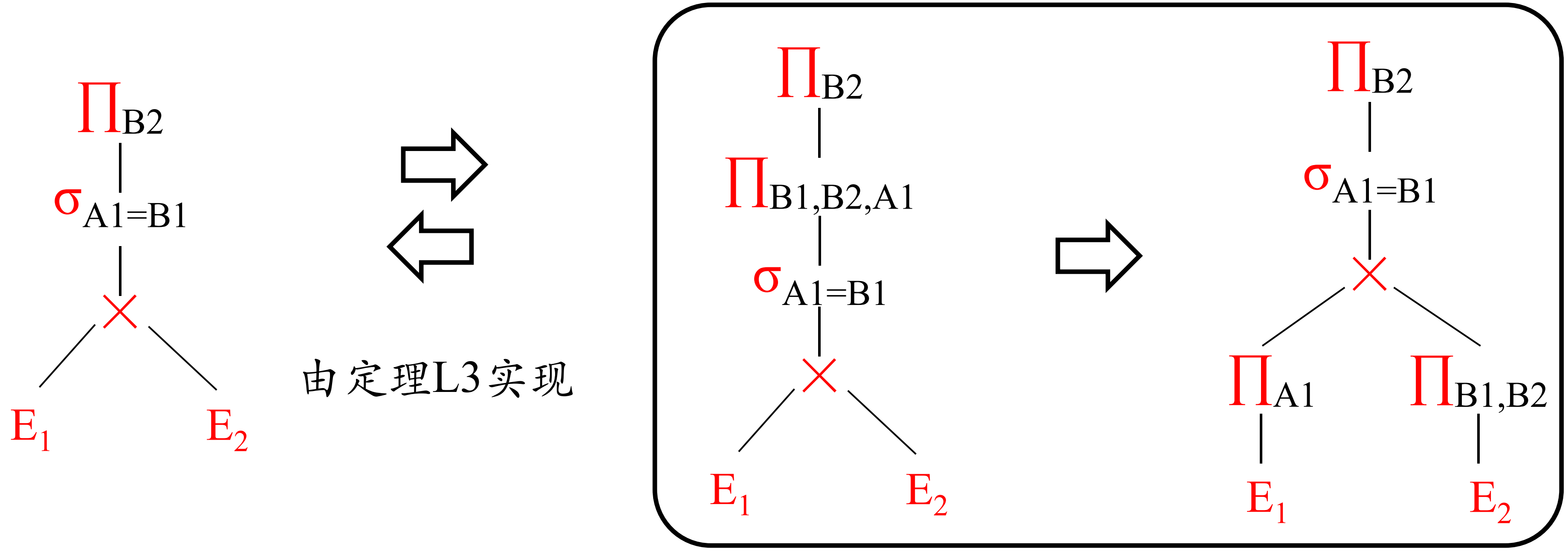
（2）关系代数操作次序交换的等价规则

定理L7：投影和积的交换律

设E_1, E_2为两关系代数表达式，$A_1, \ldots, A_n$是出现在E_1或E_2中的一些属性， 其中$B_1, \ldots, B_m$出现在E_1中，剩余的属性$C_1, \ldots, C_k$出现在E_2中， $\{A_1, \ldots, A_n\} = \{B_1, \ldots, B_m\} \cup \{C_1, \ldots, C_k\}$

则有：

$$\Pi_{A1,\ldots,An}(E_1 \times E_2) \equiv (\Pi_{B1,\ldots,Bm} E_1) \times (\Pi_{C1,\ldots,Ck} E_2)$$



2. 逻辑查询优化

（2） 关系代数操作次序交换的等价规则

定理L8：选择和并的交换律

关系代数表达式 $E = E_1 \cup E_2$，F是条件，则有：

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

注意：此定理要求E_1，E_2是并相容的

降低中间结果

定理L9：选择和差的交换律

关系代数表达式 $E = E_1 - E_2$，F是条件，则有：

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

（2）关系代数操作次序交换的等价规则

L10：投影和并的交换律

关系代数表达式 $E = E_1 \cup E_2$， $A_1, \ldots, A_n$是E中的一些属性，则有：

$$\prod_{A1,\ldots,An}(E_1 \cup E_2) \equiv (\prod_{A1,\ldots,An} E_1) \cup (\prod_{A1,\ldots,An} E_2)$$

注意：此定理要求E_1，E_2是并相容的

问：投影和集差运算有交换律吗?

$$\prod_{A1,\ldots,An}(E_1 - E_2) \equiv (\prod_{A1,\ldots,An} E_1) - (\prod_{A1,\ldots,An} E_2)$$

这个是不成立的，为什么呢?

2. 逻辑查询优化

（3）逻辑查询优化策略

根据一些规则对关系代数表达式进行等价变换（启发式优化）

- 尽可能地早做选择和投影：可使中间结果变小，节省几个数量级的执行时间
- 把选择与投影串接起来：一元运算序列可一起执行，只需对整个关系扫描一遍
- 将多个操作结合成一个：当R×S后有选择运算且其中有条件是R、S属性间比较的运算时，可将其转化为连接运算，使用效率更高的连接算法。
- 对多表连接采用特定顺序：减小中间结果的元组数，降低代价

- 执行连接运算前对关系做适当预处理：文件排序、建立临时索引等，可使两关系公共值高效联接
- 找出表达式里的公共子表达式：若公共子表达式结果不大，则预先计算，以后可读入此结果，尤当视图情况下有用

2. 逻辑查询优化

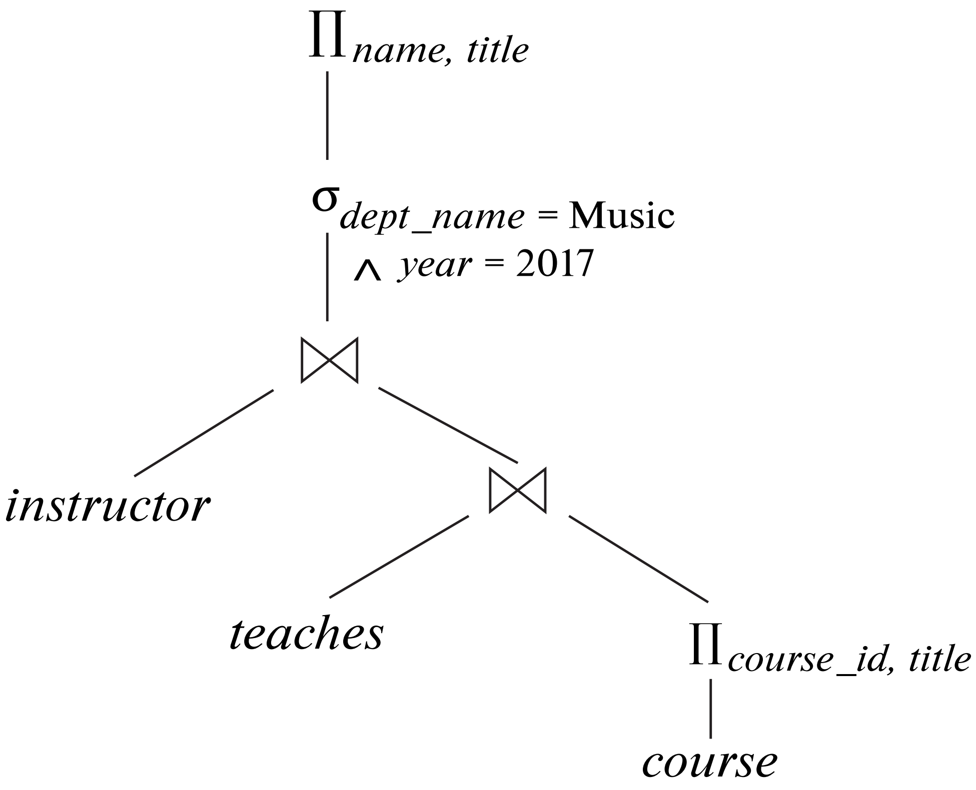
（5）逻辑查询优化示例

Instructor (ID, name, dept_name, salary)

Teaches(ID, course_id, sec_id, semester, year)

Course(course_id, title, dept_name, credits)

Q：查询所有Music系在2017年讲过课的教师名字，及每个人当年讲的所有课的名字。



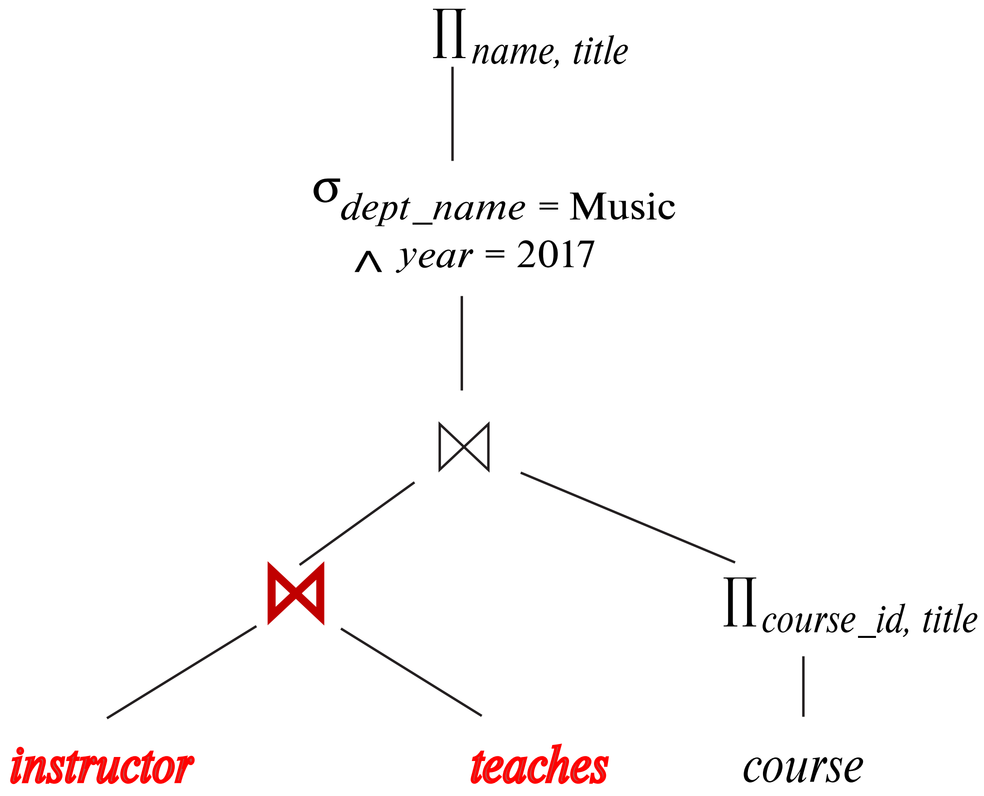
$$\Pi_{name,\ title}(\sigma_{dept\_name=\text{'Music'}\wedge year\ =\ 2017}\ (instructor \bowtie (teaches \bowtie \Pi_{course\_id,\ title}\ (course))))$$

（5）逻辑查询优化示例

Instructor (ID, name, dept_name, salary)

Teaches(ID, course_id, sec_id, semester, year)

Course(course_id, title, dept_name, credits)



Q：查询所有Music系在2017年讲过课的教师名字，及每个人当年讲的所有课的名字。

根据L2 连接操作结合律，修改连接顺序(可依据具体代价估计决定最优顺序)

$$\Pi_{name,\ title}(\sigma_{dept\_name=\text{'Music'} \wedge year = 2017}((instructor \bowtie teaches) \bowtie \Pi_{course\_id,\ title}(course)))$$

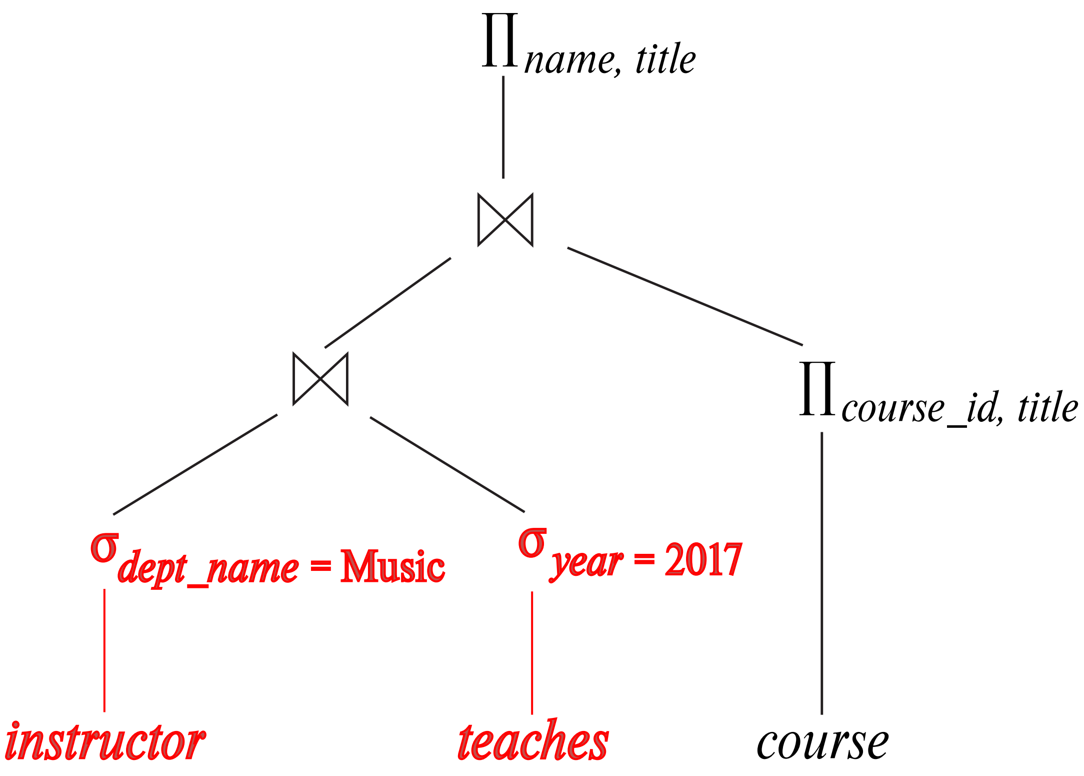
2. 逻辑查询优化

（5）逻辑查询优化示例

Instructor (ID, name, dept_name, salary)

Teaches(ID, course_id, sec_id, semester, year)

Course(course_id, title, dept_name, credits)



Q：查询所有Music系在2017年讲过课的教师名字，及每个人当年讲的所有课的名字。

根据L4, L6 将选择操作下沉到连接之前

$$\Pi_{name,\ title}((\sigma_{dept\_name=\text{'Music'}}(instructor) \bowtie \sigma_{year=2017}(teaches)) \bowtie \Pi_{course\_id,\ title}(course))$$

（5）逻辑查询优化示例

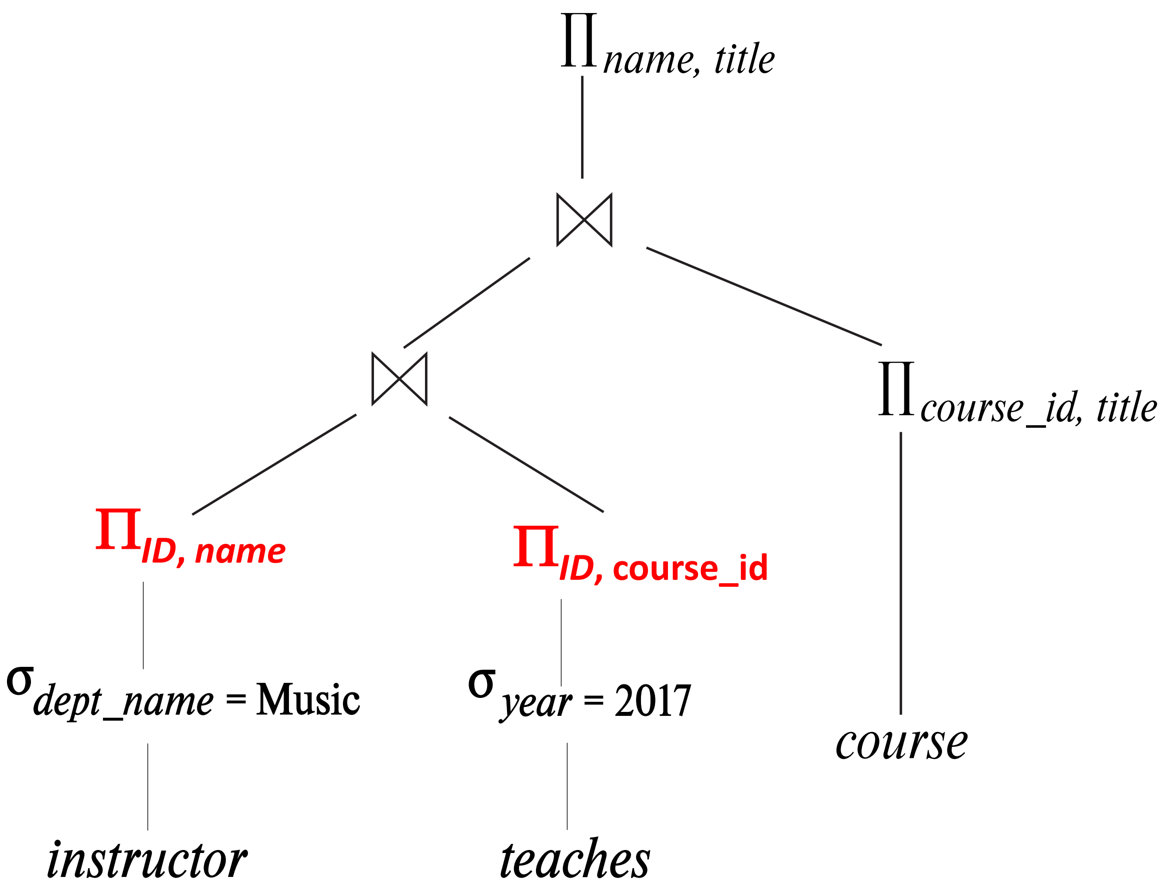
Instructor (<u>ID</u>, name, dept_name, salary)

Teaches(<u>ID</u>, <u>course_id</u>, <u>sec_id</u>, <u>semester</u>, <u>year</u>)

Course(<u>course_id</u>, title, dept_name, credits)

Q：查询所有Music系在2017年讲过课的教师名字，及每个人当年讲的所有课的名字。

根据L5在连接前先进行投影，缩小中间结果大小



$$\Pi_{name,\ title}(\Pi_{ID,\ name}(\sigma_{dept\_name=\text{'Music'}}(instructor)) \bowtie \Pi_{ID,\ course\_id}(\sigma_{year\ =\ 2017}(teaches)) \bowtie \Pi_{course\_id,\ title}(course))$$

（5）逻辑查询优化示例

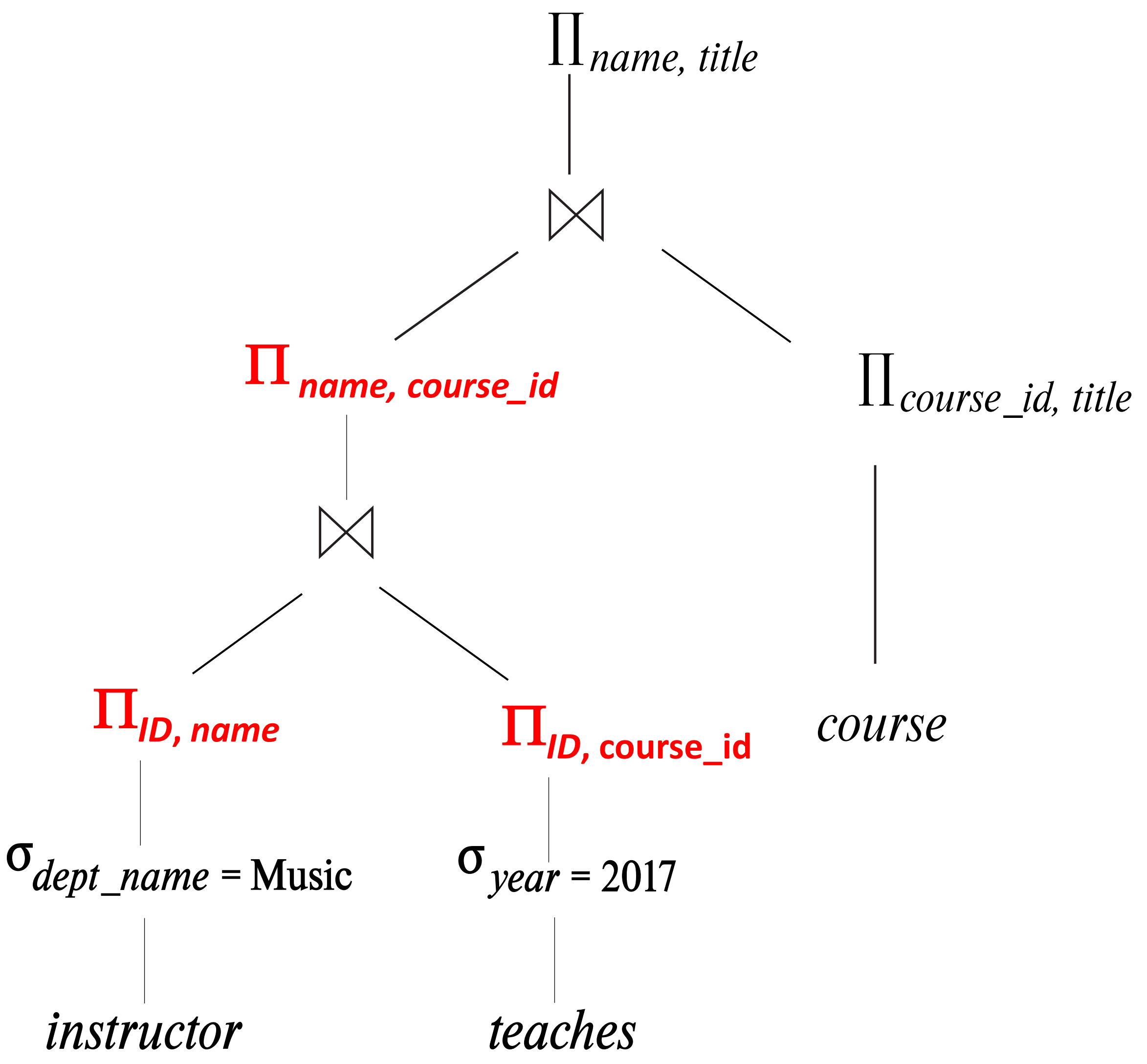
Instructor (ID, name, dept_name, salary)

Teaches(ID, course_id, sec_id, semester, year)

Course(course_id, title, dept_name, credits)

Q：查询所有Music系在2017年讲过课的教师名字，及每个人当年讲的所有课的名字。

第一次连接之后还可以继续投影到(name, course_id)上



（5）逻辑查询优化示例

考虑一图书馆的关系数据库

- BOOKS (TITLE, AUTHOR, PNAME, LC_NO)

  PNAME为出版社名，LC_NO为图书馆图书编目号
- PUBLISHERS (PNAME, PADDR, PCITY)

  PADDR为出版社地址，PCITY为出版社所在地
- BORROWERS (NAME, ADDR, CITY, CARD_NO)

  NAME为读者名，ADDR为读者所在地址，CITY为读者所在城市，CARD_NO为图书证号
- LOANS (CARD_NO, LC_NO, DATE)

  DATE为借出日期

- 为方便用户使用，定义了视图XLOANS：

  XLOANS = $\prod_S(\sigma_F(\text{LOANS} \times \text{BORROWERS} \times \text{BOOKS}))$

  S = TITLE, AUTHOR, PNAME, LC_NO, NAME, ADDR, CITY, CARD_NO, DATE

  F = (BORROWERS.CARD_NO= LOANS.CARD_NO) $\wedge$ (BOOKS.LC_NO = LOANS.LC_NO)

（5）逻辑查询优化示例 用语法树表达关系代数表达式

查询：查出1978年1月1日前被借出的所有书的书名

select Title **from** XLOAN **where** Date <= 1/1/78

⇩

$\Pi_{Title}(\sigma_{Date<='1/1/78'} XLOAN)$

⇩

语法树

Π_{Title}

$\sigma_{Date<='1/1/78'}$

XLOAN

⇨

Π_{Tilte}

σ_{F1}

Π_{S}

σ_{F}

×

× BOOKS

LOANS BORROWERS

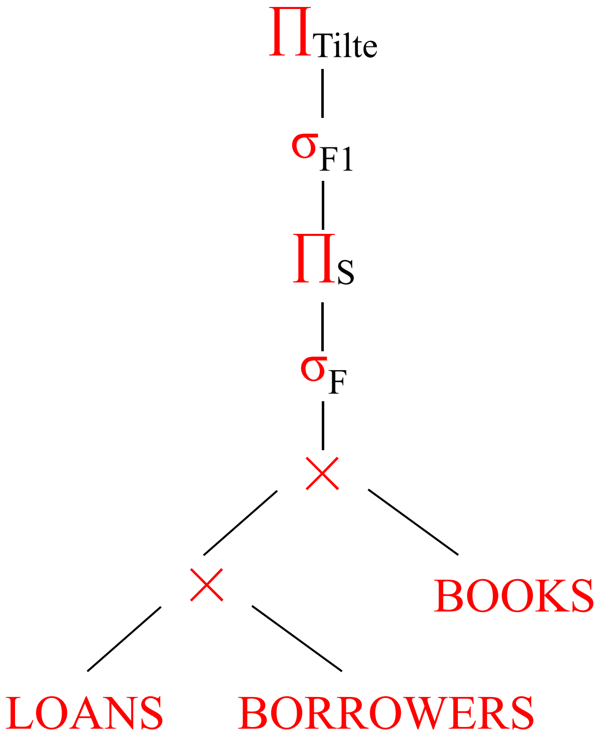
由树叶向树根反映了操作先后次序

其中：$F = F_2 \wedge F_3$；

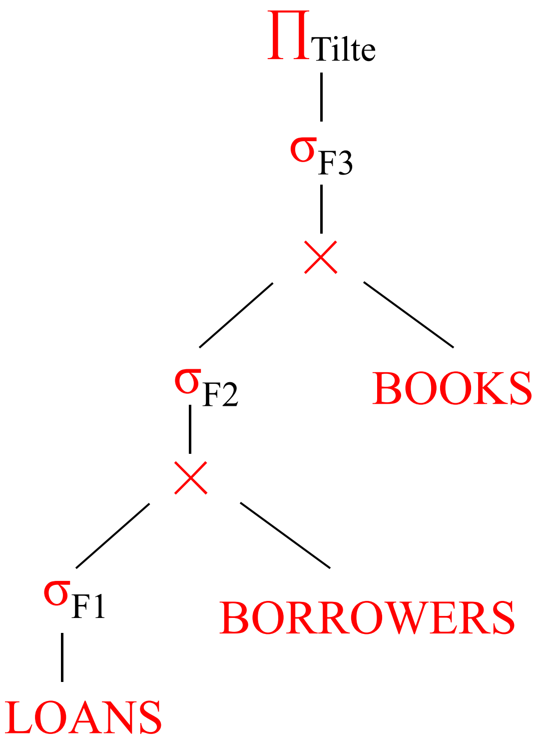
- F_1 = “DATE<=1/1/78”；
- F_2 = “BORROWERS.CARD_NO = LOANS.CARD_NO”；
- F_3 = “BOOKS.LC_NO = LOANS.LC_NO”

（5）逻辑查询优化示例

- 依据定理L4，把形如 $\sigma_{F1 \wedge F2 \wedge \ldots \wedge Fn}$ E 的选择表达式变成串接形式 $\sigma_{F1}(\sigma_{F2}(\ldots(\sigma_{Fn}E)\ldots))$
- 对每个选择，依据定理L4至L9，尽可能把它移至树的底部



- $\sigma_{F2 \wedge F3} = \sigma_{F2}(\sigma_{F3})$
- σ_{F1}通过交换移动到底部
- σ_{F2}通过交换移动到底部
- $\prod_{Title}(\prod_{S}) = \prod_{Title}$



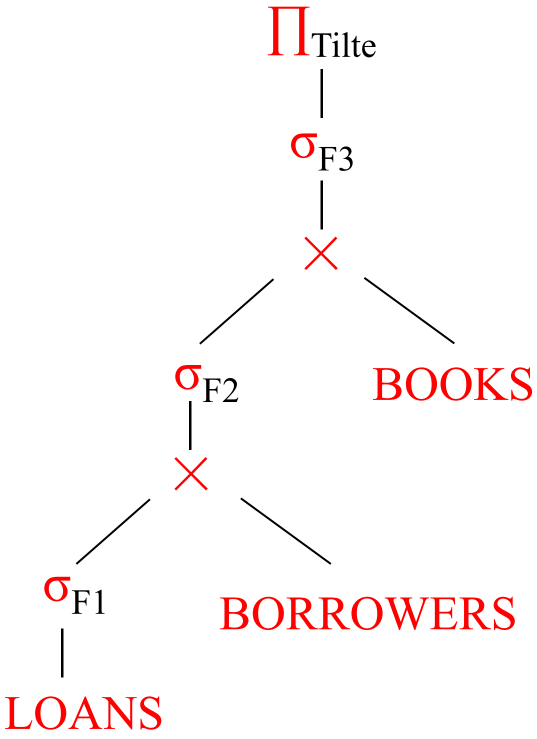
其中：$F = F_2 \wedge F_3$ ；

- F_1 = “DATE<=1/1/78” ;
- F_2 = “BORROWERS.CARD_NO = LOANS.CARD_NO” ；
- F_3 = “BOOKS.LC_NO = LOANS.LC_NO”

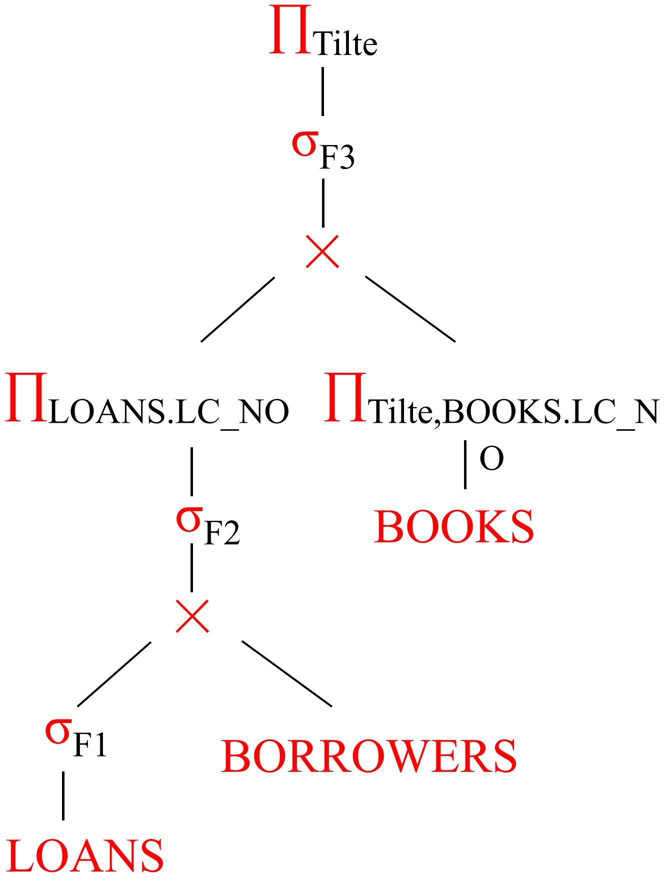
2. 逻辑查询优化

(5) 逻辑查询优化示例

- 对每个投影，依据定理L3，L7，L10和L5，尽可能把它移至树的底部。如果一个投影是对某表达式所有属性进行的，则去掉之



- $\prod_{Title} =$ $\prod_{Title}(\prod_{Title,BOOKS.LC_NO,LOANS.LC_NO})$使其包含$F_3$涉及的属性
- $\prod_{Title,BOOKS.LC_NO}$移动到底部
- $\prod_{LOANS.LC_NO}$移动到底部



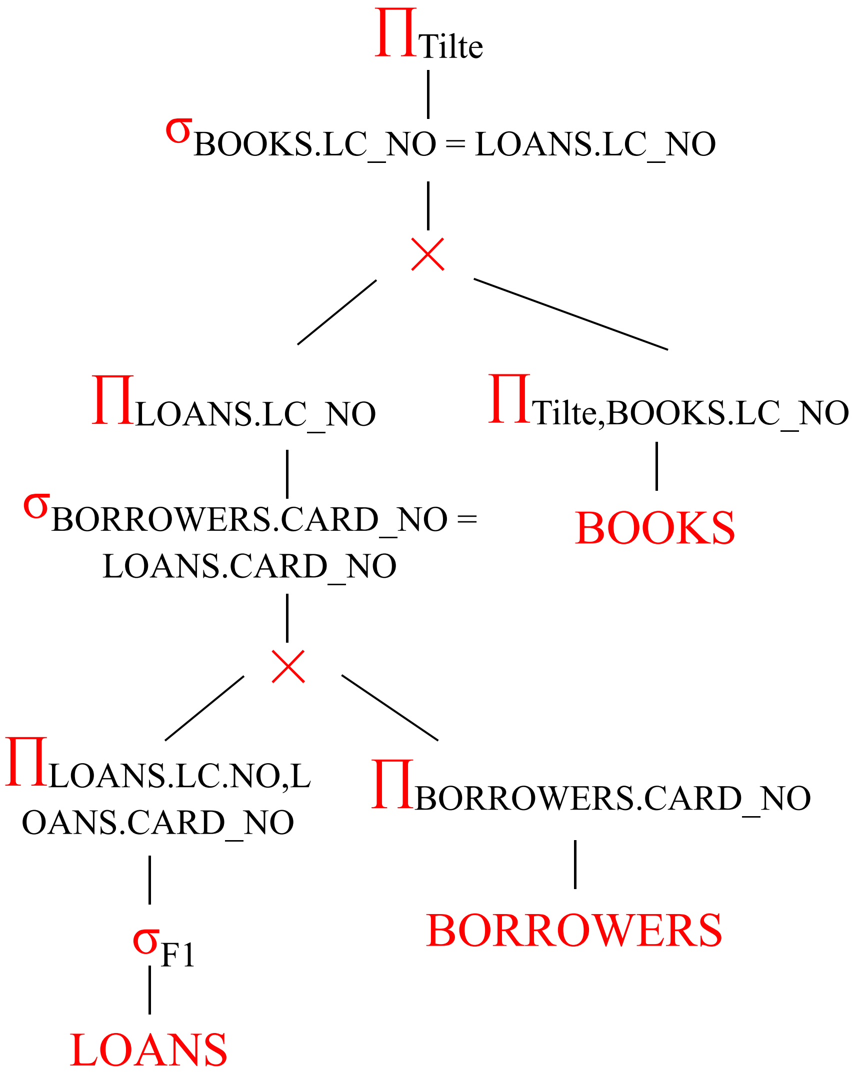
其中：$F = F_2 \wedge F_3$；

- F_1 = “DATE<=1/1/78”；
- F_2 = “BORROWERS.CARD_NO = LOANS.CARD_NO”；
- F_3 = “BOOKS.LC_NO = LOANS.LC_NO”

2. 逻辑查询优化

（5）逻辑查询优化示例

- 最终版本逻辑查询计划
- 从叶节点向根节点执行



- F_1 = “DATE<=1/1/78” ;

教学内容

1. 查询优化概述
2. 逻辑查询优化
3. <u>物理查询优化</u>
4. 代价估算

3. 物理查询优化

（1）总体思路

相同的逻辑查询方案，不同的执行算法，代价也不同

一个例子：$\sigma_{Cname=\text{“数据结构”}}$ (Course) 的执行方案

- 方案1：线性扫描
- 方案2：利用Course上的Cname B+树索引的方法

- 当条件更复杂时，可选择的方案还会更多
- 选取代价最小的方案，生成物理查询计划

$\sigma_{Cname=\text{‘数据结构’}}$

course

究竟用哪一个算法的程序来执行?为什么 如此选择?

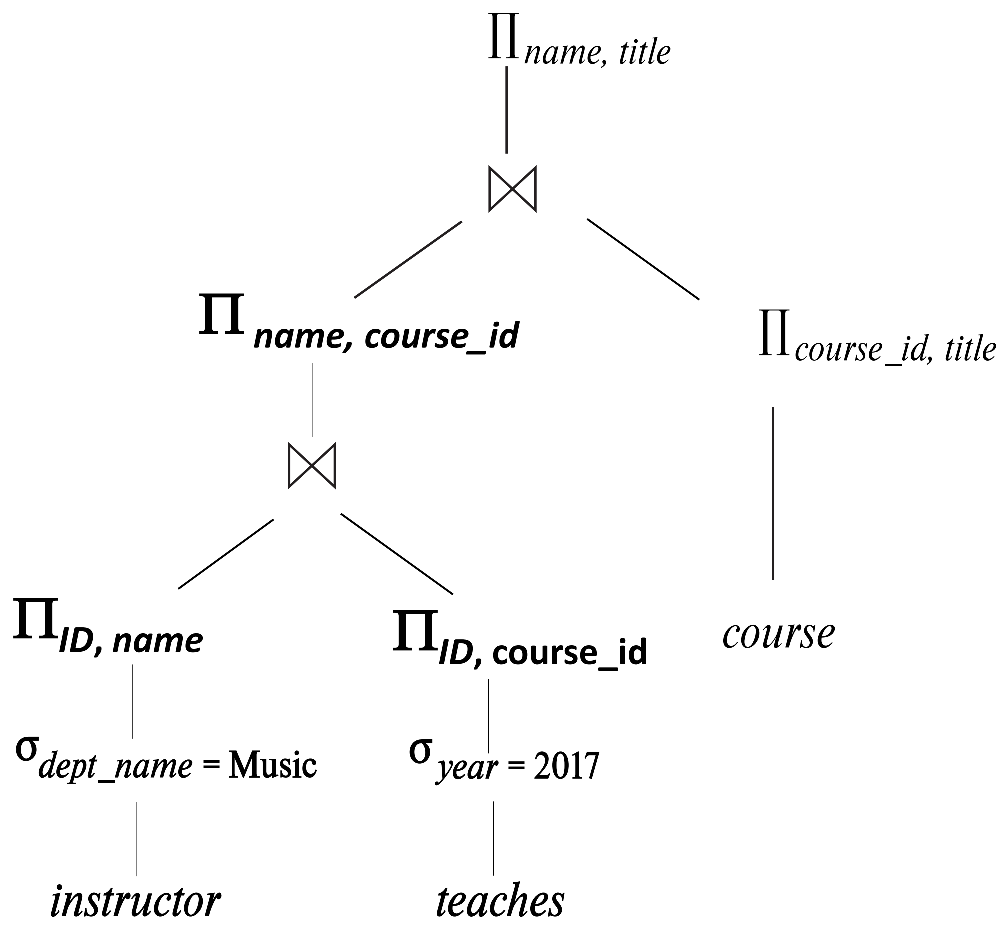
3. 物理查询优化

（1）总体思路

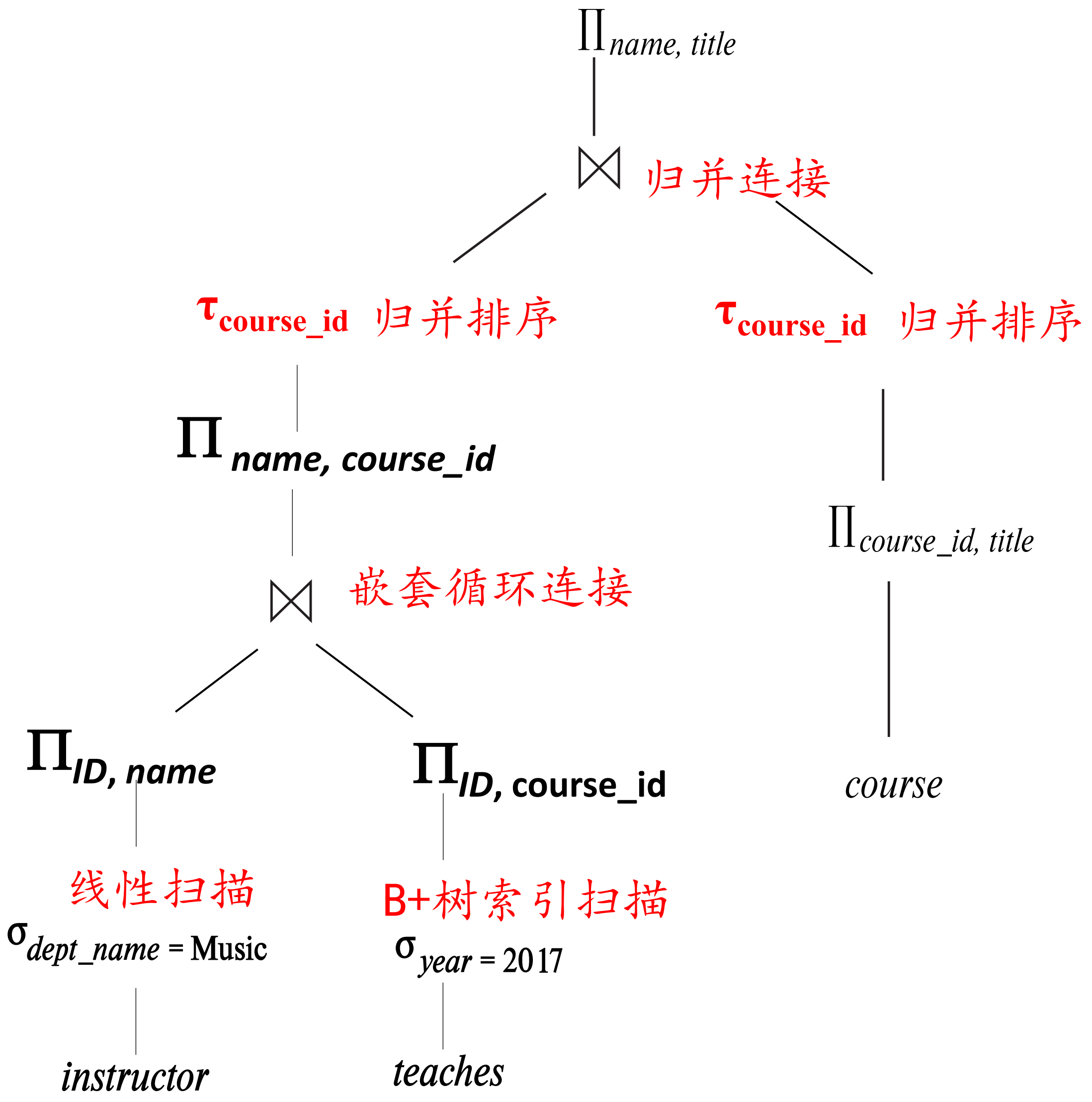
物理查询运算符

- 获取关系元组的操作
  - ✓ 全表扫描、索引扫描算法：输出未排序的元组
  - ✓ 排序扫描、排序索引扫描：输出排序后的元组
- 关系操作的各种实现算法
  - ✓ 选择：线性扫描算法、基于索引的选择算法
  - ✓ 排序：多路归并、索引排序、散列排序等
  - ✓ 笛卡尔积，连接：嵌套循环、块嵌套循环、索引连接、归并连接、散列连接等
  - ✓ 并交叉集合运算的算法
  - ✓ 其他操作$\delta(R)$, $\gamma(R)$, $\tau(R)$：去重、分组、聚集函数等的具体算法
- 迭代器构造--流水化、物化；

逻辑查询计划



物理查询计划



3. 物理查询优化

（2）基于代价的查询优化策略

- DBMS如何衡量物理查询计划的优劣
  - ✓ 衡量I/O访问次数(最重要)
  - ✓ 衡量CPU的占用时间
  - ✓ 内存使用代价(与缓冲区数目与大小的匹配)
  - ✓ 中间结果存储代价
  - ✓ 计算量(如搜索记录、合并记录、排序记录、字段值的计算等)
  - ✓ 网络通信量
  - ✓ … …

依据什么信息来计算这些方案的上述各种指标

3. 物理查询优化

（2）衡量物理查询计划

- 依据数据库的一些统计信息——存放在数据字典或系统目录中
  - ✓ T_R或T(R)：关系R的元组数目
  - ✓ B_R或B(R)：关系R的磁盘块数目
  - ✓ I_R或I(R)：关系R的每个元组的字节数
  - ✓ f_R或f(R)：R的块因子，即一块能够存储的R的元组数目
  - ✓ V(A, R)：R中属性A出现不同值的数目，即$\prod_A(R)$的数目.
  - ✓ SC(A, R)：R中属性A的选择基数，满足A上等值条件的平均记录数
  - ✓ b：每个磁盘块的字节数;
  - ✓ … …

- DBMS依据上述统计信息对DB操作的各种物理查询计划进行评估，以确定最优的计划予以执行

上述信息如何获得

3. 物理查询优化

（3）收集统计信息

- 现代DBMS可以自动或由用户/DBA手动搜集、更新统计信息。
- 一般在表建立以及增删改操作时，会更新统计信息，但不一定完全实时。
- 通常通过随机抽样等方法估计，而不是准确值，故存在误差。
- 通常在DBMS轻负载时运行

- 多数主流数据库可以设置自动更新统计信息模式
- 也可以由DBA手动进行
  - IBM DB2使用Runstats命令
  - Postgres 使用 Analyze 命令等

有了统计信息，如何进行代价估算呢?

$R_1 \bowtie R_2 \bowtie \ldots \bowtie R_n$ 的连接顺序确定问题

物理查询计划的形成:
- 理想：寻找最优的查询计划
- 现实：避免最差的查询计划

教学内容

1. 查询优化概述
2. 逻辑查询优化
3. 物理查询优化
4. <u>代价估算</u>

3. 代价估算

（1）什么是代价估算

已知表达式E中的各个关系的统计信息

- T_R或T(R)：关系R的元组数目；
- B_R或B(R)：关系R的磁盘块数目；
- I_R或I(R)：关系R的每个元组的字节数;
- f_R或f(R)：R的块因子，即一块能够存储的R的元组数目 (blocking factor)
- V(R, A)：R中属性A出现不同值的数目，即 $\prod_A(R)$ 的数目.
- SC(R, A)：R中属性A的选择基数，满足A上等值条件的平均记录数

给定一个表达式E，如何计算E的元组数目T(E)以及属性A上不同值的数目V(E, A)

- 在E实际获得之前计算T(A), V(E, A)等是很难的事情
- 因而要“估算”，代价估算

3. 代价估算

（2）代价估算

投影运算

估算一个投影 $\prod_L(R)$ 的大小

- $T(\prod_L(R)) = T(R)$
- 投影运算只是对列有所取舍，并未对行有所变化，如并未消除重复
- 投影运算并未减少行数，但可能有效地减少了存储结果关系的块数
- 例如：磁盘块大小=1024 Byte，

R的元组长度=100 Byte, 8元组/块，T(R)=10,000, 则

$B(R) = 10000/8 = 1250;$

$\prod_L(R)$的元组长度=20 Byte, 50元组/块，则

$B(\prod_L(R)) = 10000/50 = 200;$

3. 代价估算

（2）代价估算

选择运算

估算选择运算 $S = \sigma_{A=c}(R)$ 的大小

- T(S) 介于 0 to T(R)–V(R, A)+1 之间

  ——最多：A属性不同值的元组都只存在一个，剩余的都是A=c的元组

- 估计：T(S) = T(R)/V(R, A)

  ——A属性不同值的元组数假设是均匀分布的

估算选择运算 $S = \sigma_{A<c}(R)$ 的大小

- T(S) 介于 0 to T(R) 之间

  ——最多：所有元组都满足条件

- 估计：T(S) = T(R)/2

  ——假设c在A取值范围中排序的位置是均匀分布的

3. 代价估算

（2）代价估算

选择运算

估算选择运算 $S = \sigma_{A=10 \text{ AND } B<20}(R)$ 的大小

- 估计：$T(S) = T(R)/(V(R, A)*2)$

——$\sigma_{A=10 \text{ AND } B<20}(R) = \sigma_{B<20}(\sigma_{A=10}(R))$

——A=10，得出 $T(S) = T(R)/V(R, A)$;

——B<20，得出 $T(S) = T(S)/2$（按上页讲的均匀分布估计）

这里有什么假设？

估算选择运算 $S = \sigma_{C1 \text{ OR } C2}(R)$ 的大小

- 估计：$T(S) = n(1-(1-m_1/n)(1-m_2/n))$

——R有n个元组，其中有 m_1 个满足 C_1，有 m_2 个满足 C_2

——$(1-m_1/n)$ 是不满足 C_1 的那些元组，$(1-m_2/n)$ 是不满足 C_2 的那些元组

——两数之积是不在S中的那部分R的元组，1减去这个积就是属于S的那部分元组出现的概率

3. 代价估算

（2）代价估算

选择运算

估算选择运算 $S = \sigma_{A=10\ OR\ B<20}(R)$ 的大小

- 估计：$T(S) = n(1-(1-m_1/n)(1-m_2/n))$

——$n = T(R)=10000$, $V(R,A)=50$,

$m_1 = T(R)/V(R, A) = 10000/50 = 200$;

$m_2 = T(R)/2 = 10000/2 = 5000$

（有 m_1 个满足 C_1, 有 m_2 个满足 C_2,

$(1-m_1/n)(1-m_2/n)$ 不满足这个条件的元组的概率

$1- (1-m_1/n)(1-m_2/n)$ 满足这个条件的元组的概率)

——复杂估计：$T(S) = 10000*(1-(1-200/10000)(1-5000/10000)) = 5100$

——简单估计：$T(S)= T(R)/2 = 10000/2 = 5000$

（2）代价估算

连接运算

$\boxed{\text{估算连接运算 } S = R(X,Y) \bowtie S(Y,Z) \text{ 的大小}}$

- 估计：$T(S) = T(R)T(S)/\max(V(R, Y),V(S, Y))$

  ——假定$V(R, Y) >= V(S, Y)$，R中元组r和S中元组有相同Y值的概率 = $1/V(R, Y)$

  ——假定$V(R, Y) < V(S, Y)$，R中元组r和S中元组有相同Y值的概率 = $1/V(S, Y)$

  ——则，在Y上相等的概率 = $1/\max(V(R, Y),V(S, Y))$

- 例：$T(R)=10000, T(S)=50000, V(R, Y) = 500, V(S, Y)=1000$

  ——估计：$T(S)=10000*50000/1000=500000$。

最终：查询执行计划优化

- 对每个查询执行计划计算一个总代价值
- 基于代价的最优查询计划搜索是一个NP-Hard问题
- 结合前面讲的启发式优化策略减小搜索空间

- 具体执行方式：迭代器算法-流水线执行
- 含有**排序**、**去重**、**分组**等步骤的操作不能按元组流水，必须等上一步操作全结束才能开始
- 其他操作可以按元组流水，避免中间结果的存储

- 可以按流水的阶段分组执行