查询处理

查询处理

总述

内容：查询是DB的核心操作，查询优化是RDBMS的核心技术。本知识点学习RDBMS处理查询的过程。

查询处理

一、查询处理的概念

二、查询处理的步骤

一、查询处理的概念

假设数据库中有学生表S(sno,sname)和成绩表SC(sno,cno,grade)，用户想查询选了课程‘C2’的学生姓名，则需要用SQL语言编写如下查询：

```
SELECT sname
FROM S,SC
WHERE S.sno=SC.sno AND SC.cno= ‘c2’
```

RDBMS需要将用户提交的SQL查询语句转化为高效的查询执行计划，执行后返回查询结果。

查询处理是RDBMS执行查询语句的过程。

二、查询处理的步骤

（1）查询分析

（2）查询检查

（3）查询优化

（4）查询执行

二、查询处理的步骤

（1）查询分析

例：已知S(sno,sname)和SC(sno,cno,grade)

```
SELECT sname
FROM S,SC
WHERE S.sno=SC.sno AND SC.cno= ‘c2’
```

RDBMS处理查询的第一步是查询分析，查询分析的主要任务是进行语法和词法检查，判断是否符合SQL语法规则。

二、查询处理的步骤

（2）查询检查

例：已知S(sno,sname)和SC(sno,cno,grade)

```
SELECT sname
FROM S,SC
WHERE S.sno=SC.sno AND SC.cno= ‘c2’
```

RDBMS处理查询的第二步是查询检查，查询检查的主要任务是进行语义检查，判断是否符合存取权限和完整性约束。

生成查询树来表示扩展的关系代数形式。

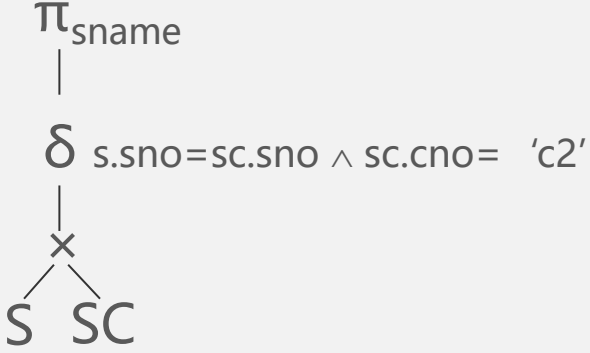
二、查询处理的步骤

（2）查询检查

例：SELECT sname
　　FROM S,SC
　　WHERE S.sno=SC.sno AND SC.cno= ‘c2’

关系代数：$\pi_{sname}(\delta_{S.sno=SC.sno \wedge cno=\ 'c2'}(S \times SC))$

查询树：



二、查询处理的步骤

（3）查询优化

例：已知S(sno,sname)和SC(sno,cno,grade)

```
SELECT sname
FROM S,SC
WHERE S.sno=SC.sno AND SC.cno= ‘c2’
```

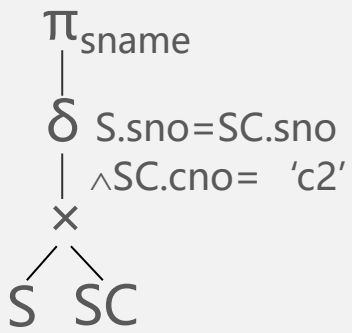
RDBMS处理查询的第三步是查询优化，查询优化即选择高效的查询处理策略，生成优化的查询树，及选择存取路径和底层算法。

二、查询处理的步骤

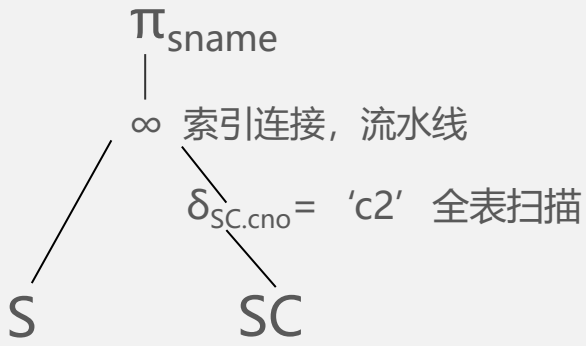
（3）查询优化

例：SELECT sname
FROM S,SC
WHERE S.sno=SC.sno AND SC.cno= ‘c2’

查询树：



优化的查询树：



二、查询处理的步骤

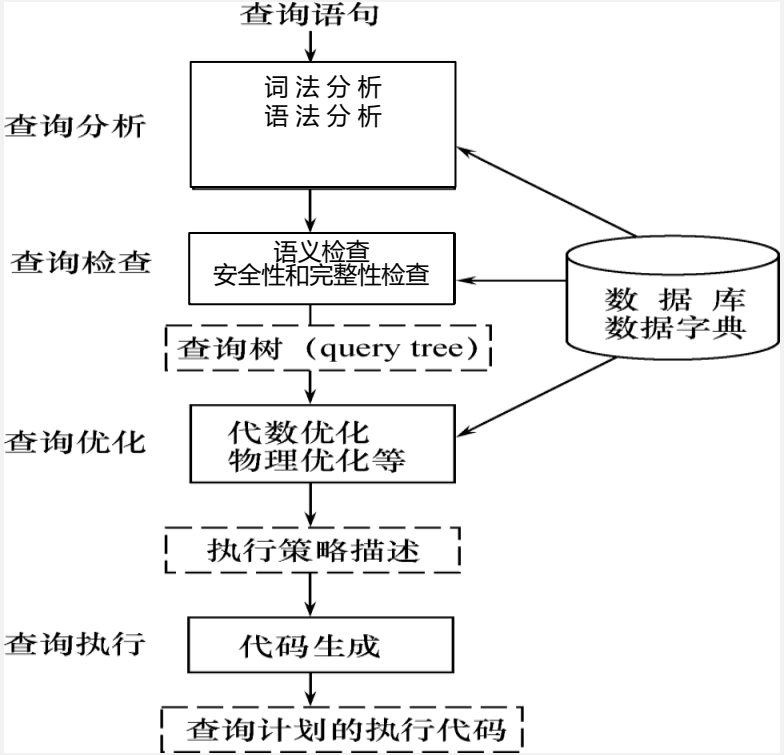
（4）查询执行

例：已知S(sno,sname)和SC(sno,cno,grade)

```
SELECT sname
FROM S,SC
WHERE S.sno=SC.sno AND SC.cno= ‘c2’
```

RDBMS处理查询的最后一步是查询执行，即生成查询计划和执行。

二、查询处理的步骤


查询语句
词法分析
语法分析
查询分析
语义检查
安全性和完整性检查
查询检查
数据库
数据字典
查询树（query tree）
代数优化
物理优化等
查询优化
执行策略描述
代码生成
查询执行
查询计划的执行代码

练习

查询处理的关键技术是（ ）。

A. 视图技术

B. 封锁技术

C. 事务处理技术

D. 查询优化技术

查询处理

解答

查询处理的关键技术是（D）。

A. 视图技术

B. 封锁技术

C. 事务处理技术

D. 查询优化技术

小结

RDBMS处理查询的过程分为查询分析、查询检查、查询优化和查询执行四个阶段。其中，查询优化是RDBMS的核心技术，是影响RDBMS性能的关键因素。


谢谢！

查询优化概述

查询优化概述

总述

内容：查询是DB的核心操作，查询优化是RDBMS的核心技术，是影响RDBMS性能的关键因素。本知识点学习查询优化的基本概念。

查询优化概述

一、查询优化的必要性

二、查询优化的定义

一、查询优化的必要性

例1：查询选了课程’ C2’ 的学生姓名。

```
SELECT sname
FROM S,SC
WHERE S.sno=SC.sno and cno= ‘c2’
```

$Q1=\pi_{sname}(\delta_{S.sno=SC.sno \wedge SC.cno= \text{‘c2’}}(S\times SC))$

$Q2=\pi_{sname}(\delta_{S.sno=SC.sno}(S\times\delta_{cno= \text{‘c2’}}(SC)))$

$Q3=\pi_{sname}(S \infty \delta_{cno= \text{‘c2’}}(SC))$

RDBMS通过代价模型计算各种查询执行策略的执行代价。

一、查询优化的必要性

在基于代价的模型中:

总代价=I/O代价+CPU代价+内存代价

I/O代价与数据库中数据的存储结构和存取方式有关。

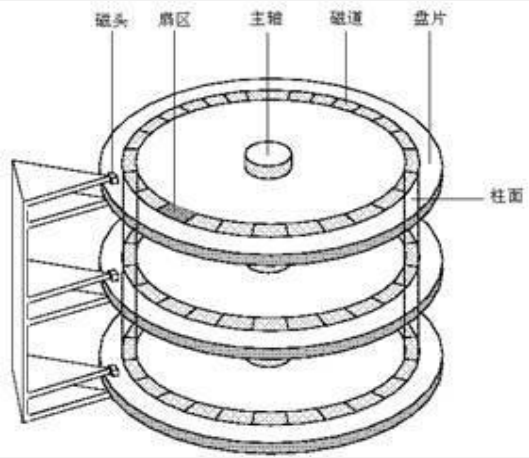
数据库是大量持久数据的集合，数据存储在硬盘上，访问数据从硬盘读入缓冲区，系统对缓存数据操作，若是修改操作，结果还需写回磁盘。

一、查询优化的必要性

在基于代价的模型中:

总代价=I/O代价+CPU代价+内存代价

读数据时间=寻磁道时间+旋转扇区时间+数据传输时间



一、查询优化的必要性

对于$Q1=\pi_{sname}(\delta_{s.sno=sc.sno \wedge sc.cno='c2'}(S\times SC))$，先做笛卡尔积，要把S的每个元组与SC的每个元组连接起来。

设S和SC的元组个数都是10000，每个物理块可存5个元组，那么S和SC各有2000块，而内存只给这个操作100块空间。

可让S的第一组99块数据装入内存，然后将SC逐块装入内存去做元组连接。

再把S的第二组99块数据装入内存，然后将SC逐块装入内存去做元组连接……，直到S表处理完为止。

一、查询优化的必要性

$Q1=\pi_{sname}(\delta_{s.sno=sc.sno \wedge sc.cno='c2'}(S\times SC))$

S和SC各有2000块。

S的每块只进内存一次，装入2000块；SC的每块进内存(2000/99)次，装入(2000/99)×2000块，因而执行S×SC的总装入块数是：

2000+（2000/99）×2000≈42400（块）

一、查询优化的必要性

$Q2=\pi_{sname}(\delta_{S.sno=SC.sno}(S\times\delta_{cno='c2'}(SC)))$

$Q3=\pi_{sname}(S\infty\delta_{cno='c2'}(SC))$

对于Q2和Q3，由于先做选择，设SC中cno= ‘c2’的元组只有5个，因此关系SC的每块只需进内存一次，则S与SC的总装入块数是4000，相当于求Q1花费时间的1/10。

可见，查询执行时选择、投影和连接的顺序不同，查询效率相差很大。

一、查询优化的必要性

在关系代数运算中，通常是先进行笛卡尔积或连接运算，再进行选择和投影；

实现查询优化则应恰当地安排选择、投影和连接的顺序，即先进行选择和投影，再做连接。

一、查询优化的必要性

为什么是系统做查询优化?

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。

- 优化器可以从数据字典中获得信息;
- 优化器可以考虑数百种执行计划;
- 优化器可以包括很多复杂的优化技术,计算各种查询执行策略的执行代价,选取代价最小的方案。

二、查询优化的定义

DBMS为了提高时空效率，减少运行时间，在对数据进行操作之前，先对用户的操作语句进行转化，以得到一串所需执行时间较少、占用空间较小的关系运算，以及较优化的存取路径，以提高系统的时空效率。

按优化的层次查询优化一般可分为代数优化和物理优化。代数优化是关系代数表达式的优化，物理优化是存取路径及底层算法的选择。

练习

下列4项中,必须进行查询优化的是 （ ）。

A. 关系数据库

B. 网状数据库

C. 层次数据库

D. 非关系模型

解答

下列4项中,必须进行查询优化的是 （ A ）。

A. 关系数据库

B. 网状数据库

C. 层次数据库

D. 非关系模型

小结

关系系统的查询优化既是RDBMS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。

查询优化按优化的层次可分为代数优化和物理优化。


谢谢！

关系代数等价变换

关系代数等价变换

总述

内容：按优化的层次查询优化一般可分为代数优化和物理优化，代数优化是关系代数表达式的优化。本知识点学习代数优化中关系代数表达式等价变换的规则。

关系代数等价变换

一、等价变换的定义

二、常用的等价变换规则

关系代数等价变换

一、等价变换的定义

$E1=\pi_{sname}(\delta_{S.sno=SC.sno \wedge SC.cno='c2'}(S\times SC))$

$E2=\pi_{sname}(\delta_{S.sno=SC.sno}(S\times\delta_{cno='c2'}(SC)))$

两个关系代数表达式等价是指用同样的关系实例代替两个表达式中相应关系时所得到的结果是一样的。

两个关系代数表达式E1和E2等价写成：E1≡E2。

二、常用的等价变换规则

（1）笛卡尔积和联接的交换律

$$E1 \times E2 \equiv E2 \times E1$$

$$E1 \infty E2 \equiv E2 \infty E1$$

$$E1 \underset{F}{\infty} E2 \equiv E2 \underset{F}{\infty} E1$$

二、常用的等价变换规则

（2）笛卡尔积和联接的结合律

$$(E1\times E2)\times E3\equiv E1\times(E2\times E3)$$

$$(E1\infty E2)\infty E3\equiv E1\infty(E2\infty E3)$$

$$(E1\underset{F}{\infty}E2)\underset{F}{\infty}E3\equiv E1\underset{F}{\infty}(E2\underset{F}{\infty}E3)$$

二、常用的等价变换规则

（3）投影的串接定律

已知{ A1,A2,...,AN }是{ B1,B2,...,BM }的子集，

则：

$$\pi_{A1,A2,...,AN}(\pi_{B1,B2,...,BM}(E)) \equiv \pi_{A1,A2,...,AN}(E)$$

二、常用的等价变换规则

（4）选择的串接定律

选择条件可以合并，一次就可检查所有条件。

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

二、常用的等价变换规则

（5）选择和投影的交换律

若F只涉及属性{A1,A2,...,AN},则:

$\pi_{A1,A2,...,AN}(\delta_F(E)) \equiv \delta_F(\pi_{A1,A2,...,AN}(E))$

若F中有不属于{ A1,A2,...,AN }的属性{ B1,B2,...,BM }，则有更一般的规则:

$\pi_{A1,A2,...,AN}(\delta_F(E)) \equiv$

$\pi_{A1,A2,...,AN}(\delta_F(\pi_{A1,A2,...,AN,B1,B2,...,BM}(E)))$

二、常用的等价变换规则

（6）选择和笛卡尔积的交换律

若F中的属性都是E1中的，则

$$\delta_F(E1\times E2) \equiv \delta_F(E1)\times E2 ;$$

若F=F1∧F2，且F1只涉及E1中的属性，F2只涉及E2中的属性，则：

$$\delta_F(E1\times E2)\equiv\delta_{F1}(E1)\times\delta_{F2}(E2);$$

若F=F1∧F2,且F1只涉及E1中的属性,F2涉及E1和E2中的属性，则：

$$\delta_F(E1\times E2)\equiv\delta_{F2}(\delta_{F1}(E1)\times E2)。$$

二、常用的等价变换规则

（7）选择和并的交换律

E=E1∪E2,E1和E2有相同的属性集，则

$\delta_F(E1 \cup E2) \equiv \delta_F(E1) \cup \delta_F(E2)$

（8）选择和差的交换律

E=E1-E2,E1和E2有相同的属性集，则

$\delta_F(E1 - E2) \equiv \delta_F(E1) - \delta_F(E2)$

二、常用的等价变换规则

（9）投影和笛卡尔积的交换律

若{ A1,A2,...,AN }是E1的属性,

{ B1,B2,...,BM }是E2的属性，则：

$$(\pi_{A1,A2,...,AN,B1,B2,...,BM}(E1 \times E2) \equiv$$

$$\pi_{A1,A2,...,AN}(E1) \times \pi_{B1,B2,...,BM}(E2)$$

二、常用的等价变换规则

（10）投影和并的交换律

若E1、E2有相同的属性集，则：

$$(\pi_{A1,A2,\ldots,AN}(E1 \cup E2) \equiv \pi_{A1,A2,\ldots,AN}(E1) \cup \pi_{A1,A2,\ldots,AN}(E2)$$

二、常用的等价变换规则

1-2: 连接、笛卡尔积的交换律、结合律;

3: 合并或分解投影运算;

4: 合并或分解选择运算;

5-8: 选择运算与其他运算交换;

5，9，10: 投影运算与其他运算交换。

关系代数等价变换

练习

下面的等价变换表达式中，不一定成立的是（　　）。

A. $E1 \times E2 \equiv E2 \times E1$

B. $\delta_{F1}(\delta_{F2}(E)) \equiv \delta_{F1 \wedge F2}(E))$

C. $\pi_{A1,A2,\ldots,AN}(\pi_{B1,B2,\ldots,BM}(E)) \equiv \pi_{A1,A2,\ldots,AN}(E)$

D. $\delta_F(E1 \times E2) \equiv \delta_F(E1) \times E2$

关系代数等价变换

解答

下面的等价变换表达式中，不一定成立的是（D）。

A. $E1 \times E2 \equiv E2 \times E1$

B. $\delta_{F1}(\delta_{F2}(E)) \equiv \delta_{F1 \wedge F2}(E))$

C. $\pi_{A1,A2,\ldots,AN}(\pi_{B1,B2,\ldots,BM}(E)) \equiv \pi_{A1,A2,\ldots,AN}(E)$

D. $\delta_F(E1 \times E2) \equiv \delta_F(E1) \times E2$

小结

DBMS处理查询时将SQL语句转化为关系代数表达式，再按一定规则改变表达式中操作的顺序，对表达式进行变换，从而得到与原表达式等价但执行效率更高的表达式。


谢谢！

代数优化策略

代数优化策略

总述

内容：按优化的层次查询优化一般可分为代数优化和物理优化，代数优化是关系代数表达式的优化。本知识点学习代数优化中关系代数表达式优化的启发式规则。

代数优化策略

一、基于启发式规则的代数优化

二、典型的启发式规则

代数优化策略

一、基于启发式规则的代数优化

DBMS处理查询时将SQL语句转化为关系代数表达式，再按一定规则改变表达式中操作的顺序，对表达式进行变换，从而得到与原表达式等价但执行效率更高的表达式，即代数优化。

最好找出所有等价表达式，然后对每个表达式进行开销估计，找出“时空”开销最低者来执行。

但这样做效率低，也不现实。一般采用启发式规则找到一个等价的较优的表达式。

二、典型的启发式规则

(1) 提早执行选择运算；

(2) 同一关系的选择和投影运算可同时进行（减少Ⅰ／O次数）；

(3) 把投影运算同其前或后的双目运算结合；

(4) 合并笛卡尔积与其后的选择为连接运算；连接特别是等值连接运算要比同一关系上的笛卡尔积省很多时间。

二、典型的启发式规则

(5) 简化多余运算，比如有空关系的运算；

表达式	简化为	表达式	简化为
$A \cup \Phi$	A	$\delta_F \Phi$	Φ
$A \cap \Phi$	Φ	$A \times \Phi$	A
$A - \Phi$	A	$A \cup A$	A
$\Phi - A$	Φ	$A \cap A$	A
$\pi_r \Phi$	Φ	$A - A$	Φ

二、典型的启发式规则

(6) 找出公共子表达式，对它只执行一次，将其结果存到临时关系中（内存或外存）；

- 重复出现的子表达式的结果集规模不大但计算量大，临时写入外存中再读入所需的时间比重新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可以提高效率。
- SELECT * FROM S_G WHERE cname= ‘DB’

二、典型的启发式规则

(7) 适当地对关系文件进行预处理：对文件进行排序和在连接属性上建立索引。

sno	cno	grade
S2	C1	76
S3	C3	75
S1	C3	86
S2	C2	89
S1	C4	77
S4	C4	80
S1	C2	78
S1	C1	90

sno	cno	grade
S1	C1	90
S1	C2	78
S1	C3	86
S1	C4	77
S2	C1	76
S2	C2	89
S3	C3	75
S4	C4	80

练习

在下列关于代数优化的启发式规则中最基本的一条是（　）。

A. 选择运算尽可能先做

B. 投影运算尽可能先做

C. 利用空操作对表达式简化

D. 计算公共子表达式并将结果存于中间文件中

代数优化策略

解答

在下列关于代数优化的启发式规则中最基本的一条是（A）。

A. 选择运算尽可能先做

B. 投影运算尽可能先做

C. 利用空操作对表达式简化

D. 计算公共子表达式并将结果存于中间文件中

小结

代数优化一般采用启发式规则找到一个等价的较优的表达式。其中选择运算尽可能先做，是最重要、最基本的一条优化策略。


谢谢！

代数优化算法

代数优化算法

总述

内容：按优化的层次查询优化一般可分为代数优化和物理优化。学习代数优化中遵循启发式规则并应用等价变换公式对关系代数表达式进行优化的优化算法。

代数优化算法

一、算法描述

二、优化过程示例

代数优化算法

一、算法描述

输入：一个关系表达式的查询树；

输出：该表达式的优化的查询树。

算法：

（1）利用选择的串接定律把形如

$\delta_{F1\wedge F2\wedge\ldots\wedge Fn}(E)$变换为$\delta_{F1}(\delta_{F2}(\ldots(\delta_{Fn}(E))\ldots))$；

（2）对每个选择，尽可能地把它移向叶子端点；

代数优化算法

一、算法描述

选择向叶子端点移动的常用规则

操作	表达式	转换后的表达式
∪	$\delta_F(A \cup B)$	$\delta_F(A) \cup \delta_F(B)$
∩	$\delta_F(A \cap B)$	$\delta_F(A) \cap \delta_F(B)$
-	$\delta_F(A - B)$	$\delta_F(A) - \delta_F(B)$
×	$\delta_F(A \times B)$	$\delta_{F3}(\delta_{F1}(A) \times \delta_{F2}(B))$
π	$\delta_F(\pi_r(A))$	$\pi_r(\delta_F(A))$

F=F1∧F2∧F3， F1仅与A有关，F2仅与B有关，F3与A和B有关。

一、算法描述

（3）对每个投影，利用下面的规则尽可能地把它移向叶子端点；

投影向叶子端点移动的常用规则：

操作	表达式	转换后的表达式
π	$\pi_A(\pi_B(E))$	$\pi_A(E)$
δ	$\delta_F(\pi_A(E))$	$\pi_A(\delta_F(E))$
∪	$\pi_A(E1 \cup E2)$	$\pi_A(E1) \cup \pi_A(E2)$
×	$\pi_{A,B}(E1 \times E2)$	$\pi_A(E1) \times \pi_B(E2)$

一、算法描述

（4）利用下面的规则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，使多个选择和投影能同时执行；

操作	表达式	转换后的表达式
δ	$\delta_{F1\wedge F2}(E))$	$\delta_{F1}(\delta_{F2}(E))$
π	$\pi_A(\pi_B(E))$	$\pi_A(E)$
δ，π	$\delta_F(\pi_A(E))$	$\pi_A(\delta_F(E))$

一、算法描述

（5）把上述得到的查询树的内结点分组，

- 每个二元运算结点(×，∞，∪，-)与其直接祖先的一元运算结点（6，π）分为一组，如果它的子孙结点一直到叶结点都是一元运算符，则也并入该组；
- 但是如果二元运算是笛卡尔积，而且后面不是与它组合成等值联接的选择时，则不能将选择与这个二元运算组成同一组；

（6）生成一个程序，每组结点的计算是程序中的一步，各步的顺序是任意的，只要保证每组均在其子孙之后执行。

二、优化过程示例

例：

学号—sno
姓名—sname
年龄—age
性别—sex
课程号—cno
课程名—cname
教师名—tname
成绩—grade

关系模式集

学生关系模式 S(sno,sname,age,sex)

课程关系模式 C(cno,cname,tname)

选课关系模式 SC(sno,cno,grade)

二、优化过程示例

检索至少学习刘利老师所授一门课的女同学的学号和姓名。

SELECT S.sno,sname FROM S,SC,C
WHERE tname= ‘刘利’ AND sex= ‘女’ AND
S.sno=SC.sno AND SC.cno=C.cno

关系代数表达式为：

$$\pi_{sno,sname}(\delta_{tname=\text{‘刘利’} \wedge sex=\text{‘女’} \wedge S.sno=SC.sno \wedge SC.cno=C.cno}(S\times SC\times C))$$

令tname= ‘刘利’ 为条件F1，sex= ‘女’ 为条件F2
S.sno=SC.sno 为条件F3，SC.cno=C.cno为条件F4

$$\pi_{sno,sname}(\delta_{F1\wedge F2\wedge F3\wedge F4}(S\times SC\times C))$$

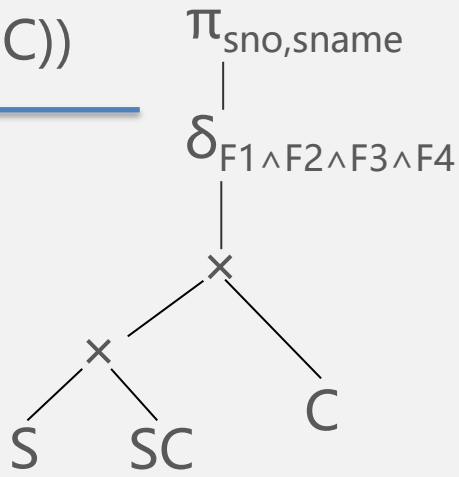
二、优化过程示例

（1）画出初始的查询树

SELECT S.sno,sname FROM S,SC,C
WHERE tname= ‘刘利’ AND sex= ‘女’ AND
S.sno=SC.sno AND SC.cno=C.cno

$\pi_{sno,sname}(\delta_{F1 \wedge F2 \wedge F3 \wedge F4}(S \times SC \times C))$

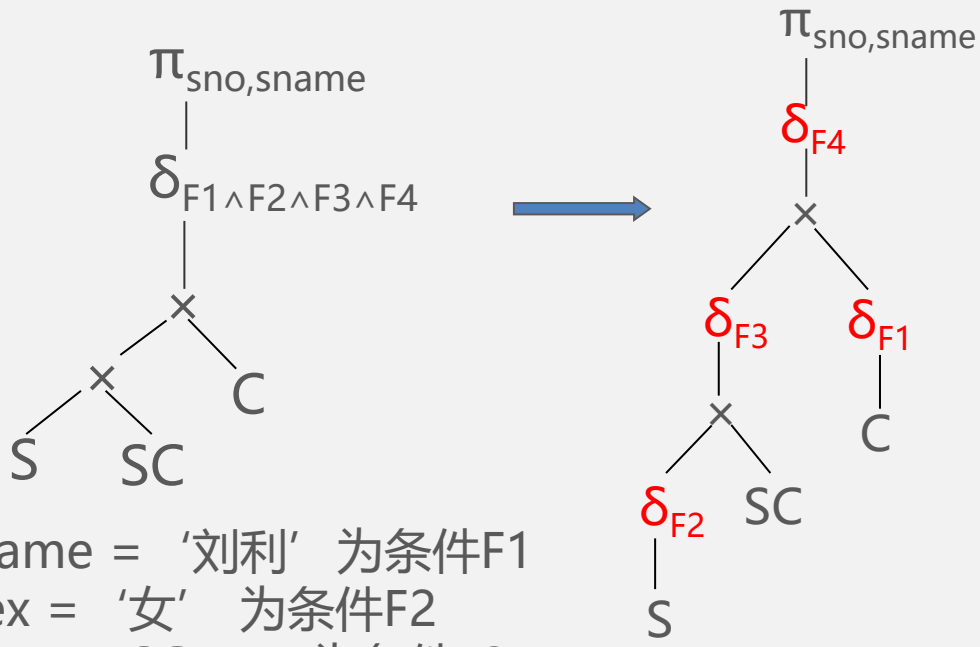
令tname = ‘刘利’ 为条件F1
sex = ‘女’ 为条件F2
S. sno=SC. sno 为条件F3
SC. cno=C. cno为条件F4



代数优化算法

二、优化过程示例

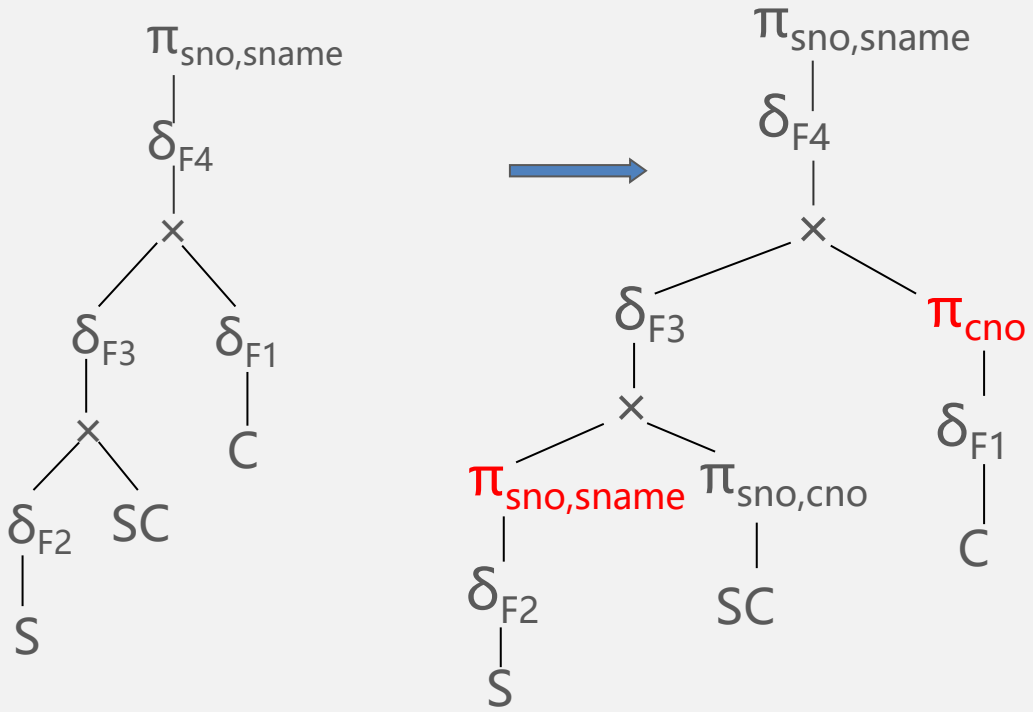
（2）将选择条件移向叶子端点



令tname = ‘刘利’ 为条件F1
sex = ‘女’ 为条件F2
S. sno=SC. sno 为条件F3
SC. cno=C. cno为条件F4

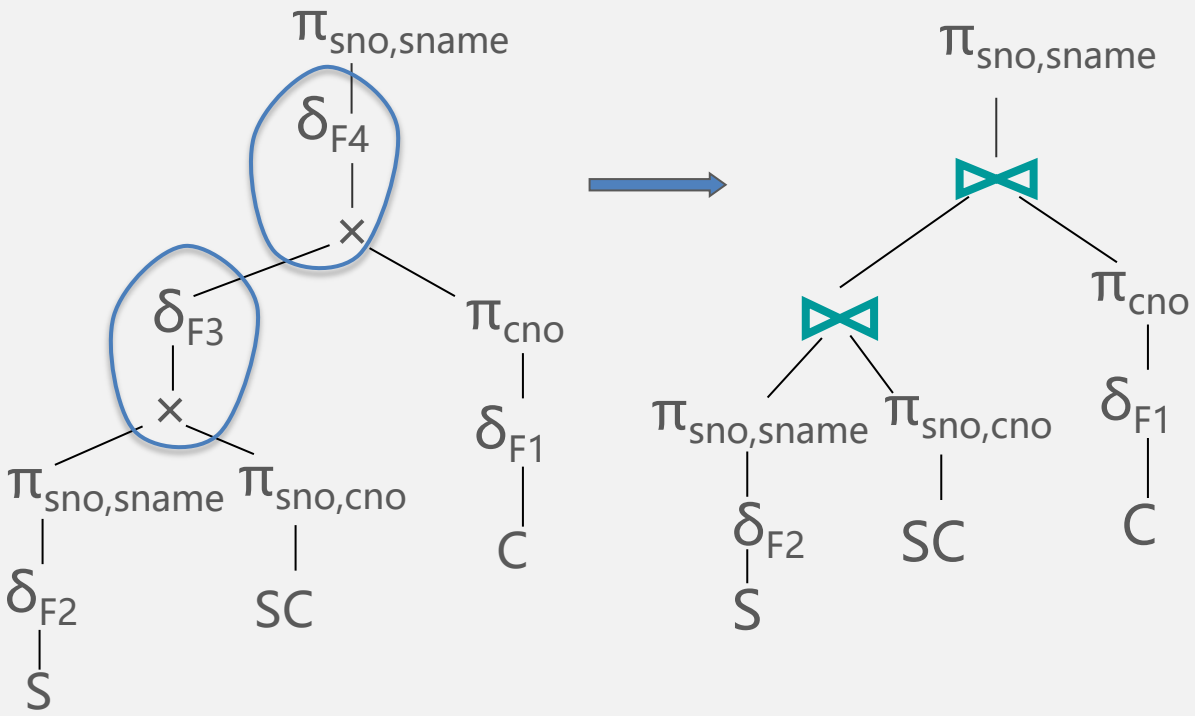
二、优化过程示例

（3）在连接前先做投影操作将无关的属性去掉



二、优化过程示例

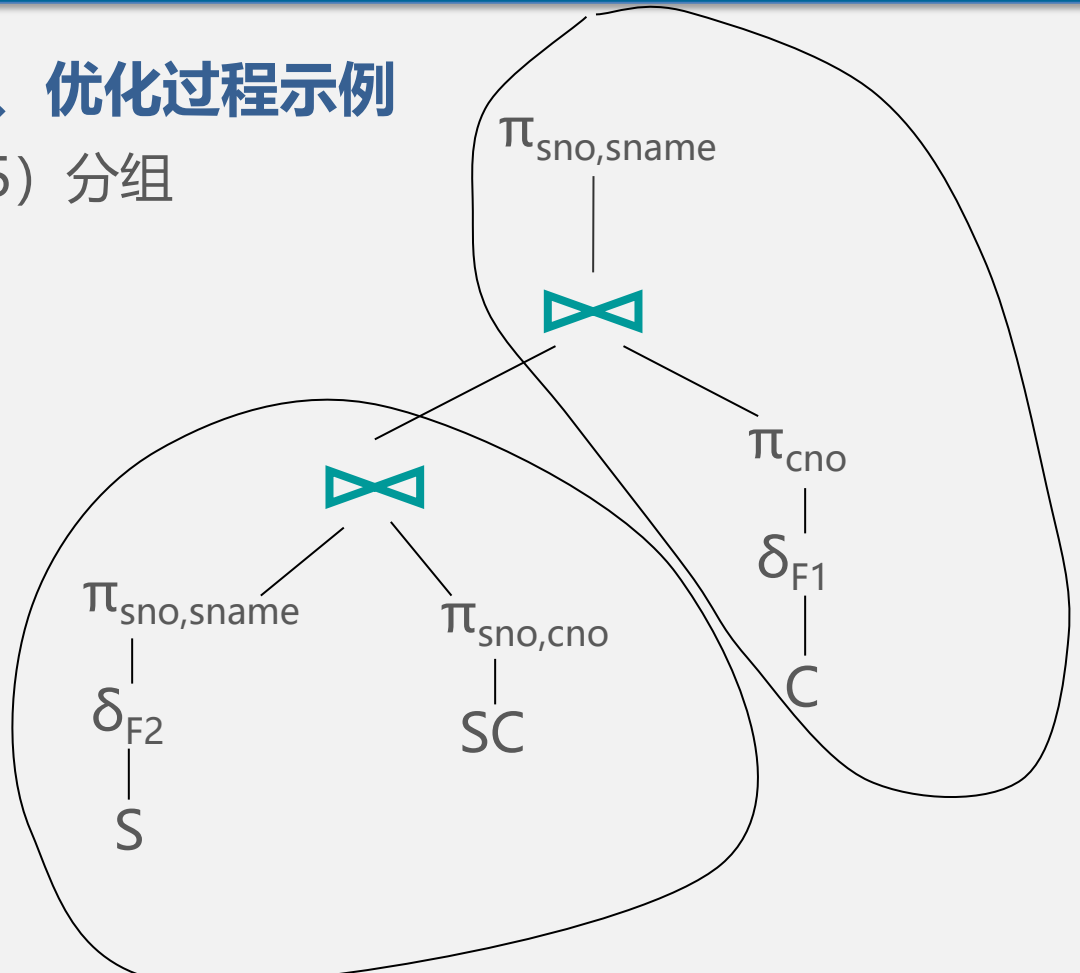
（4）将笛卡尔积和它之后的选择合并为连接



代数优化算法

二、优化过程示例

（5）分组



练习

判断对错。

任何情况下将选择操作下推到叶子节点都是一条有效的优化规则。（　）

代数优化算法

解答

任何情况下将选择操作下推到叶子节点都是一条有效的优化规则。（错）

假设S表在学号上有索引，性别上无索引，记录多，男生多；SC表记录少。求有选课的男生学号。

(1) $\pi_{sno}(\delta_{sex='男'}(S) \infty SC)$

(2) $\pi_{sno}(\delta_{sex='男'}(SC \infty S))$

学号	姓名	性别
s1	王一	男
s2	王二	男
s3	王三	男
……		

学号	课程号	成绩
s1	c1	88
s2	c1	79

小结

利用等价变换规则和启发式规则可以得到优化算法，即将一个关系表达式的语法树转化成优化的语法树形式。

启发式规则在大部分情况下适用，但不是每种情况都是最好的规则。


谢谢！

物理优化概述

物理优化概述

总述

内容：按优化的层次查询优化一般可分为代数优化和物理优化，物理优化是存取路径及底层算法的选择。本知识点学习物理优化的基本概念。

物理优化概述

一、物理优化定义

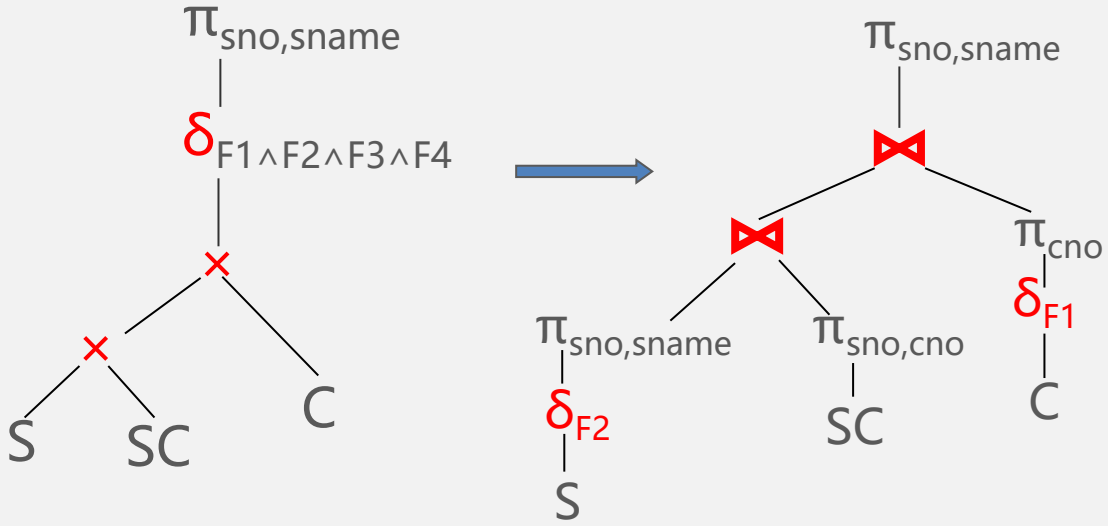
二、基于启发规则的优化

三、基于代价估算的优化

物理优化概述

一、物理优化定义

代数优化是改变查询语句中操作的次序。



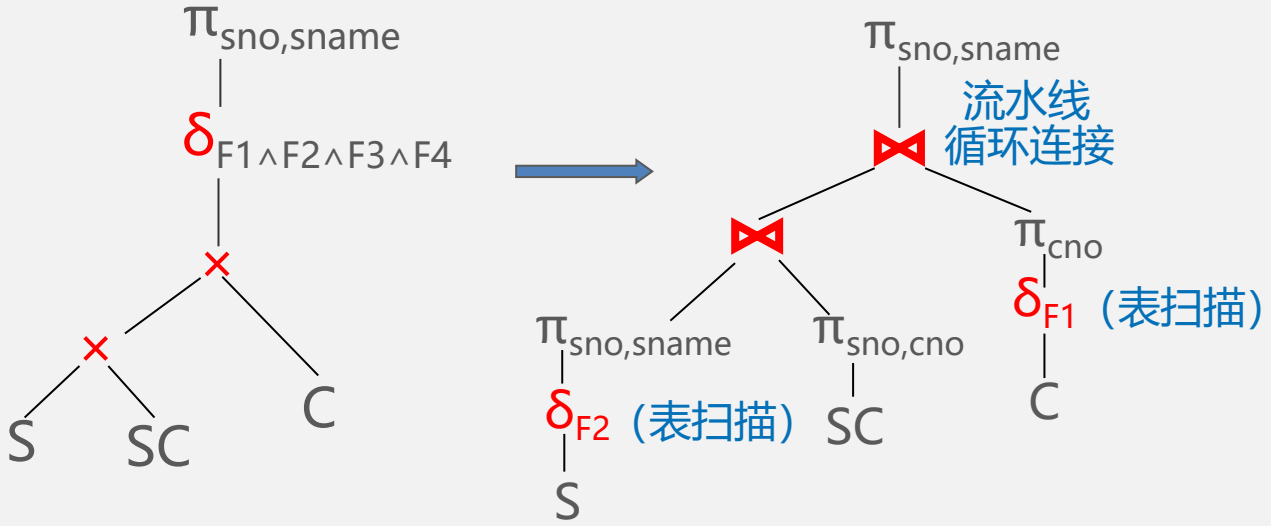
tname = ‘刘利’ 为条件F1, sex = ‘女’ 为条件F2

S. sno=SC. sno 为条件F3, SC. cno=C. cno为条件F4

物理优化概述

一、物理优化定义

物理优化是存取路径及底层算法的选择。



tname = ‘刘利’ 为条件F1, sex = ‘女’ 为条件F2

S. sno=SC. sno 为条件F3, SC. cno=C. cno为条件F4

物理优化概述

一、物理优化定义

物理优化的策略：

- 基于规则的启发式优化
- 基于代价估算的优化
- 基于语义的优化

物理优化的步骤：

先使用启发式规则，选取若干较优的候选执行方案，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。

二、基于启发规则的优化

每种关系运算都有其算法的实现，如投影、选择、连接、集合运算、聚集函数等。

比如，投影运算的关键是去重复。

- 排序，在DB中都是外排序
- 散列

三、基于代价估算的优化

启发式规则是定性的选择，比较粗糙，实现简单且优化本身代价较小，适合解释执行的系统，优化开销包含在查询总开销中 。

在编译执行的系统中，一次优化编译可多次执行，查询优化和查询执行是分开的，可以采用基于代价的优化。

三、基于代价估算的优化

基于代价估算的优化与DB状态有关，需要利用数据字典中的统计信息，主要有以下三类信息：

（1）基本表的元组总数、元组长度和占用块数

（2）基本表的每个列不同值的个数、最大值、最小值

（3）索引层数、叶结点数、不同索引值的个数

优化器利用这些统计信息就可以计算全表扫描、索引扫描、各种连接方法等操作的代价了。

练习

下面关于物理优化的论述，正确的是（　）。

A. 物理优化总能选出最优的方案

B. 物理优化无法像代数优化一样使用启发式规则

C. 在任何情况下启发式规则都是最好的规则

D. 基于代价估算的优化方法需要利用数据字典中的统计信息

解答

下面关于物理优化的论述，正确的是（D）。

A. 物理优化总能选出最优的方案

B. 物理优化无法像代数优化一样使用启发式规则

C. 在任何情况下启发式规则都是最好的规则

D. 基于代价估算的优化方法需要利用数据字典中的统计信息

小结

仅仅进行代数优化是不够的，物理优化就是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。

物理优化通常采用基于规则的启发式优化和基于代价估算的优化相结合的策略，即先使用启发式规则，选取若干较优的候选执行方案，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。


谢谢！

选择操作实现方法

选择操作实现方法

总述

按优化的层次查询优化一般可分为代数优化和物理优化。本知识点学习物理优化中DBMS对选择操作的实现方法。

选择操作实现方法

一、选择操作的实现方法

二、选择操作的启发式规则

选择操作实现方法

一、选择操作的实现方法

每种关系运算都有其算法的实现，如投影、选择、连接、集合运算、聚集函数等。

其中选择算法较复杂，包括等值和范围查询，方法有全表扫描（向前，向后）和索引（散列，B+树）。

一、选择操作的实现方法

（1）全表扫描

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。

选择率（满足条件的元组占全表的比例）较低时效率低。

一、选择操作的实现方法

（2）索引（散列，B+树）

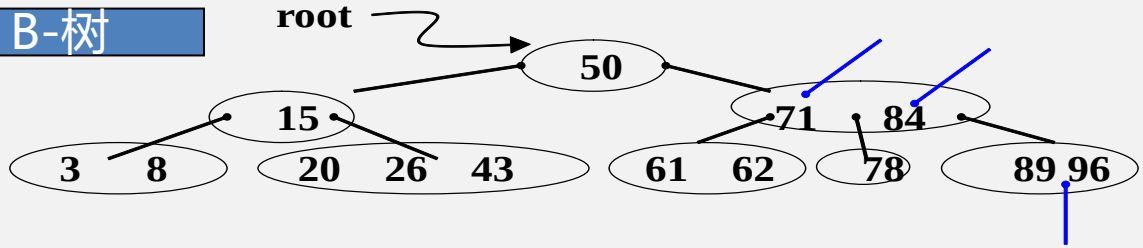
如果选择条件中的属性上有索引，可以通过索引先找到满足条件的元组主码或元组指针，通过元组指针直接在查询的基本表中找到元组。

其中，B+树基于多级索引，支持等值查找和范围查找。

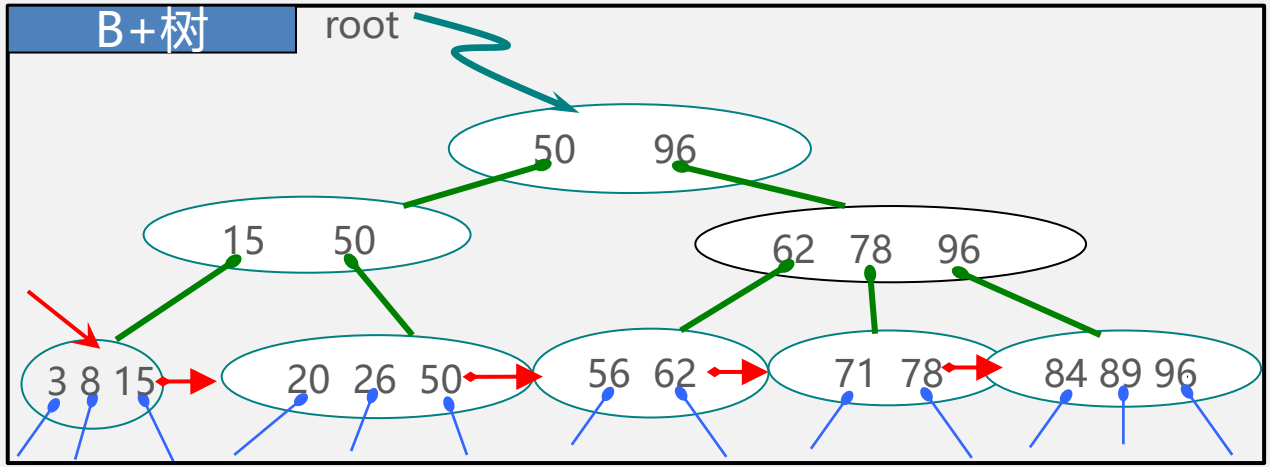
选择操作实现方法

一、选择操作的实现方法

B-树



B+树



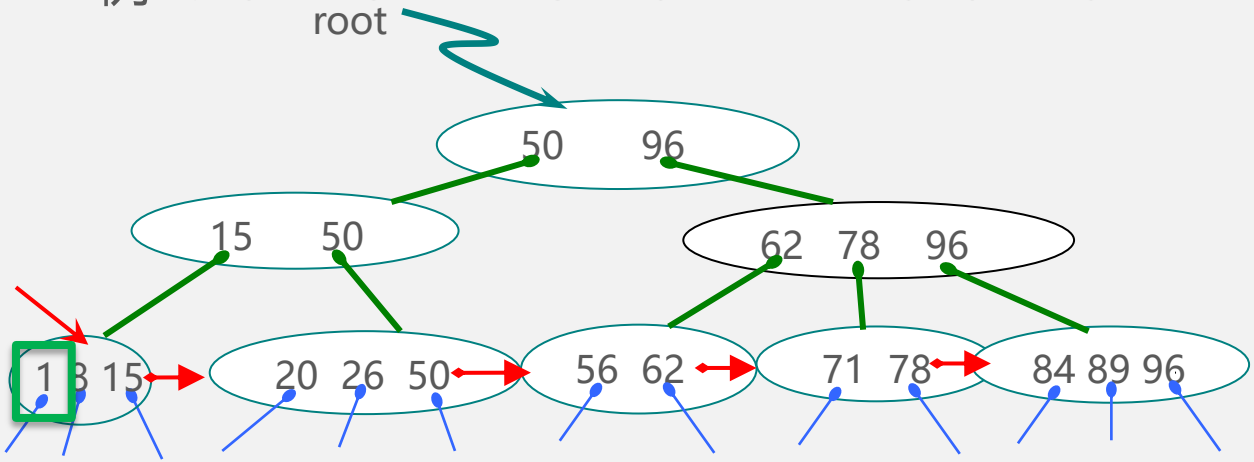
选择操作实现方法

二、选择操作的启发式规则

（1）对于小关系，使用全表顺序扫描；

（2）对于选择条件是“主键=值”的查询，选择主键索引；

例1：SELECT * FROM S WHERE sno= ‘s1’



选择操作实现方法

二、选择操作的启发式规则

（3）对于选择条件是“非主属性=值”的查询，

若该属性上有索引且结果比例较小时使用索引扫描，

否则全表顺序扫描；

例2：SELECT * FROM S WHERE age=18

选择操作实现方法

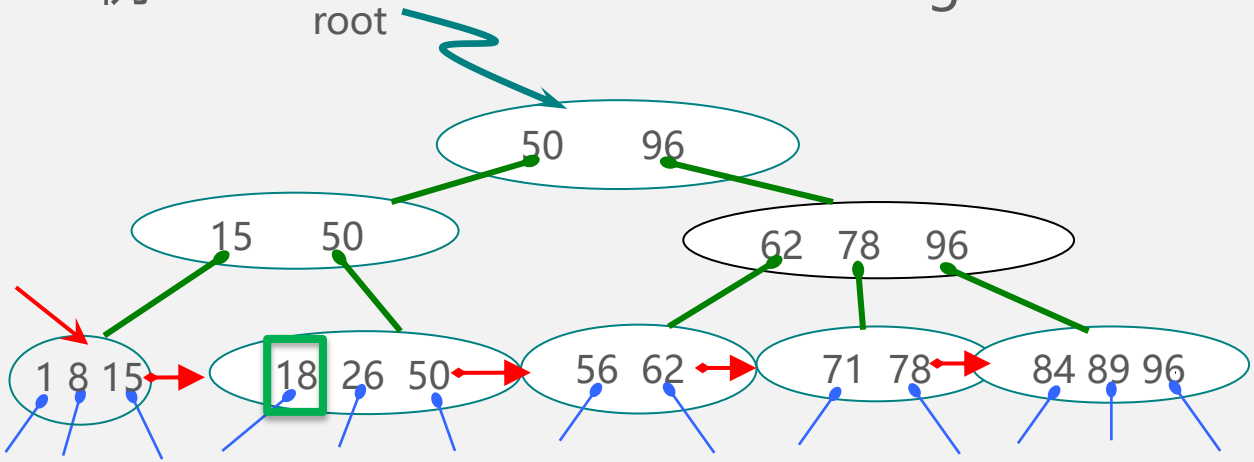
二、选择操作的启发式规则

（4）对于非等值查询或范围查询，

若该属性上有索引且结果比例较小时使用索引扫描，

否则全表顺序扫描；

例3：SELECT * FROM S WHERE age<18



选择操作实现方法

二、选择操作的启发式规则

（5）对于“或”运算，一般使用 全表顺序扫描；

（6）对于“与”运算，

按优先级顺序使用组合索引，单个索引求交集和全表顺序扫描。

选择操作实现方法

二、选择操作的启发式规则

例4：SELECT * FROM S
WHERE age>18 AND sex= ‘男’

方法1：若age和sex上有组合索引，使用组合索引；

方法2：若age和sex上都有单个索引，利用索引求得两组元组指针，再求交集，再到S表中检索；

方法3：若age和sex上都有单个索引，利用索引求得其中一组元组指针，例如先求出age>18的元组指针，利用元组指针直接到S表中检索，检查是否满足第二个条件（如sex= ‘男’），满足则输出结果。

选择操作实现方法

练习

S（sno,sname,age ,sex）和C（cno,cname，tname)，已知在S的sno和age上、C的cno上有B+树索引，说明下列查询语句较优的处理方法。

(1) SELECT * FROM S WHERE sex= ‘女’

(2) SELECT * FROM C WHERE cno= ‘c1’

(3) SELECT * FROM S WHERE age<20
and sex= ‘女’

选择操作实现方法

解答

S（sno,sname,age ,sex）和C（cno,cname，tname），已知在S的sno和age上、C的cno上有B+树索引，说明下列查询语句较优的处理方法。

(1) SELECT * FROM S WHERE sex= ‘女’
全表扫描

(2) SELECT * FROM C WHERE cno= ‘c1’
主键索引

(3) SELECT * FROM S WHERE age<20
and sex= ‘女’ age上的索引或全表扫描

选择操作实现方法

小结

每种关系运算都有其算法的实现，选择算法包括等值和范围查询。B+树基于多级索引，支持等值查找和范围查找。散列方法支持等值查找，不适合范围查找。


谢谢！

连接操作实现方法

连接操作实现方法

总述

按优化的层次查询优化一般可分为代数优化和物理优化。本知识点学习物理优化中DBMS对等值连接操作的实现方法。

连接操作实现方法

一、嵌套循环

二、排序合并连接

三、索引连接

四、哈希连接

连接操作实现方法

一、嵌套循环(NESTED LOOP)

例1：SELECT * FROM S，SC　WHERE S.sno=SC.sno ；

（1）对外层循环(S)的每一个元组，检索内层循环(SC)中的每一个元组;

（2）检查这两个元组在连接属性sno上是否相等;

（3）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 。

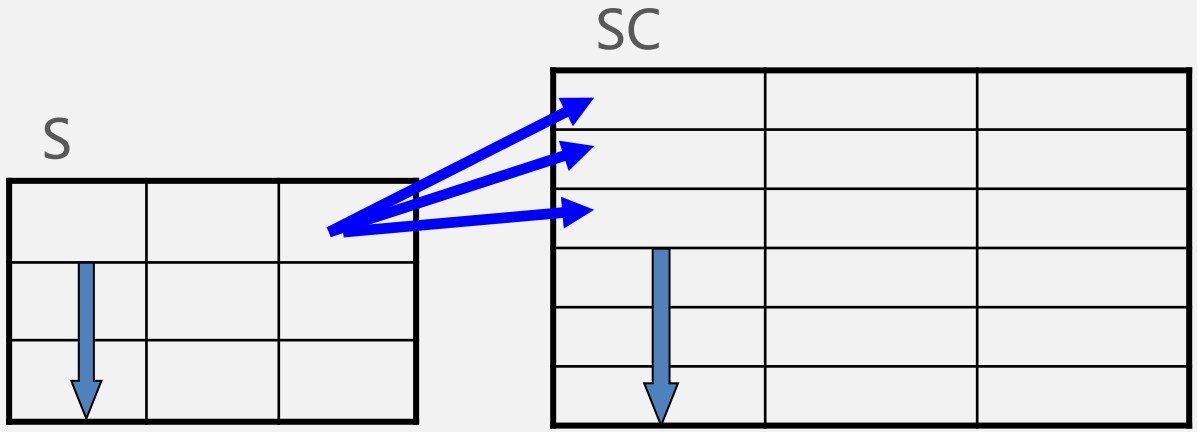
连接操作实现方法

一、嵌套循环

选择较小（占用块数较少）的表做外表。

如S表占用块数B_S，SC表占用块数B_{SC}，连接操作使用的内存缓冲区为K，则一般的做法是：

分配K-1块给外表（如S）,则嵌套循环方法存取的块数为$B_S+B_S/（K-1）\times B_{SC}$。



二、排序合并连接(SORT-MERGE JOIN)

例1：SELECT * FROM S，SC　WHERE S.sno=SC.sno；

（1）如果连接的表没有排好序，先对S表和SC表按连接属性sno排序；

（2）取S表中第一个sno，依次扫描SC表中具有相同sno的元组；

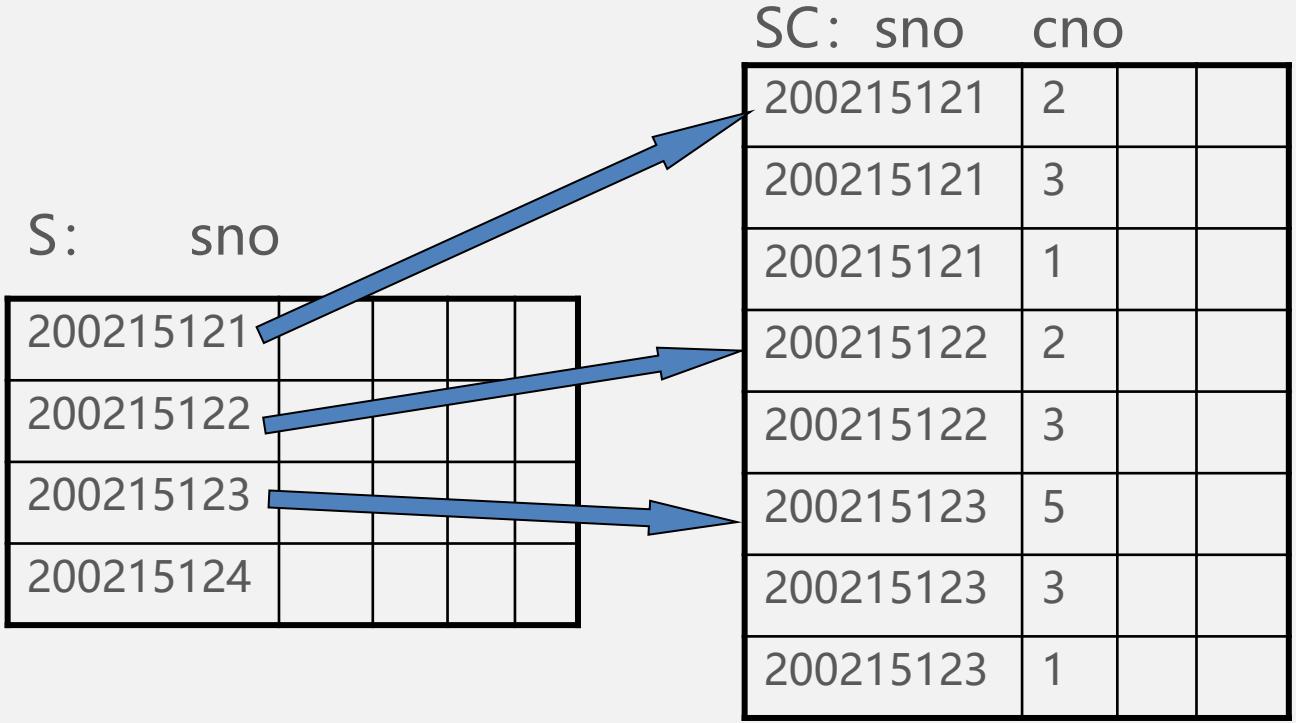
（3）当扫描到sno不相同的第一个SC元组时，返回S表扫描它的下一个元组，再扫描SC表中具有相同sno的元组，把它们连接起来；

（4）重复上述步骤直到S表扫描完。

连接操作实现方法

二、排序合并连接

对于2个大表，加上排序，总的时间一般仍会大大减少。



三、索引连接(INDEX JOIN)

例1：SELECT * FROM S，SC　WHERE S.sno=SC.sno；

（1）在SC表上建立属性sno的索引，如果原来没有该索引；

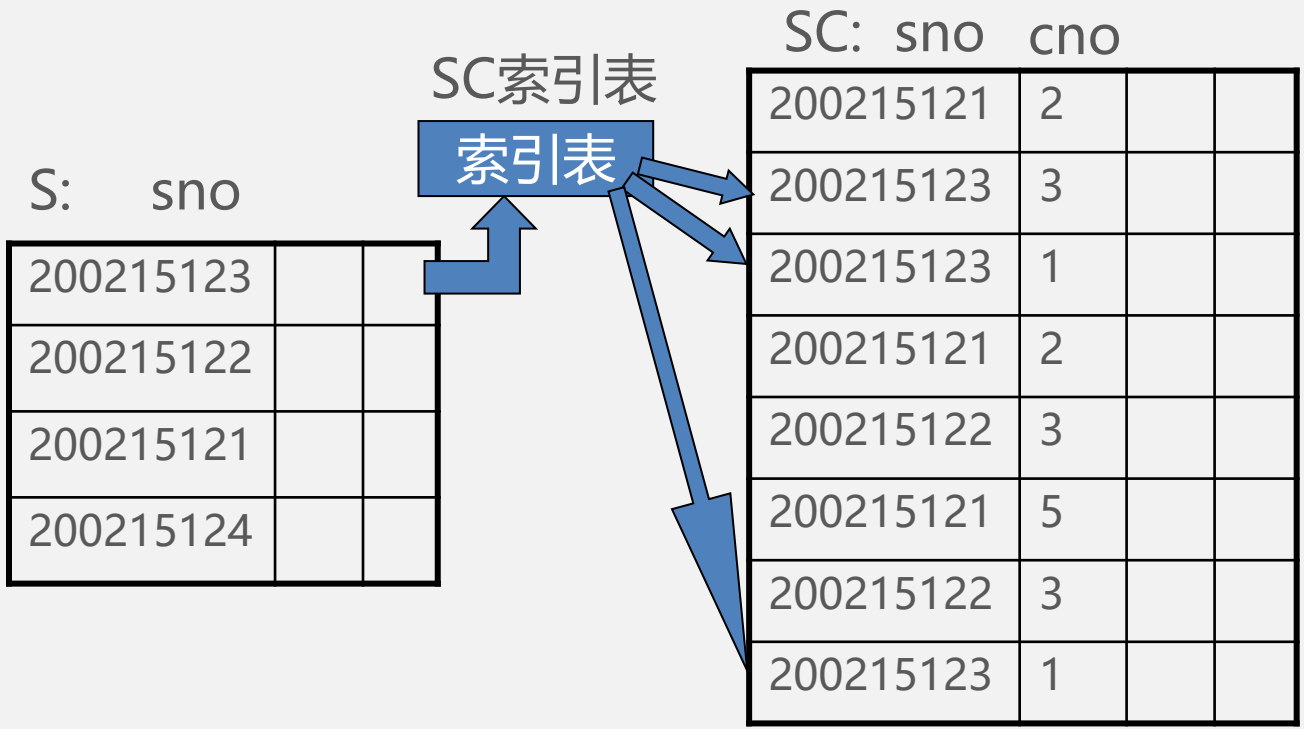
（2）对S中每一个元组，由sno值通过SC的索引查找相应的SC元组；

（3）把这些SC元组和S元组连接起来；

（4）循环执行2和3步，直到S表中的元组处理完为止。

连接操作实现方法

三、索引连接



连接操作实现方法

四、哈希连接（HASH JOIN）

例1：SELECT * FROM S，SC　WHERE S.sno=SC.sno；

（1）把连接属性作为HASH码，用同一个HASH函数把S和SC中的元组散列到同一个HASH文件中；

（2）划分阶段：对小表(比如S)进行一遍处理，把它的元组按HASH函数分散到HASH表的桶中；

（3）试探(连接)阶段：对另一个表(SC)进行一遍处理，把SC的元组散列到HASH桶中，并与桶中所有来自S并与之相匹配的元组连接起来 。

连接操作实现方法

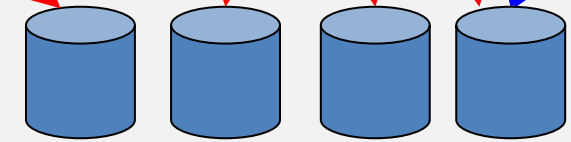
四、哈希连接

SC: sno cno

200215121	2	
200215123	3	
200215123	1	
200215121	2	
200215122	3	
200215121	5	
200215122	3	
200215123	1	

S: sno

200215123		
200215122		
200215121		
200215124		



连接操作实现方法

练习

如果两个表都已经按照连接属性排序，则优先选用的连接方法是（　）。

A. 排序合并

B. 索引连接

C. 哈希连接

D. 嵌套循环

连接操作实现方法

解答

如果两个表都已经按照连接属性排序，则优先选用的连接方法是（A）。

A. 排序合并

B. 索引连接

C. 哈希连接

D. 嵌套循环

连接操作实现方法

小结

连接操作有4种基本实现方法，可以利用下面的启发式规则进行选择：

（1）如果两个表都已经按照连接属性排序，选择排序合并方法；

（2）如果一个表在连接属性上有索引，选择索引连接方法；

（3）如果上述两个规则都不适用，其中一个表较小，选择哈希连接方法；

（4）最后是嵌套循环方法，并选择其中较小的表做外表。


谢谢！