厦门大学计算机科学系本科生课程

《数据库系统原理》

扫一扫访问班级网站
支持手机浏览

第9章 数据库查询优化（2017版）

林子雨

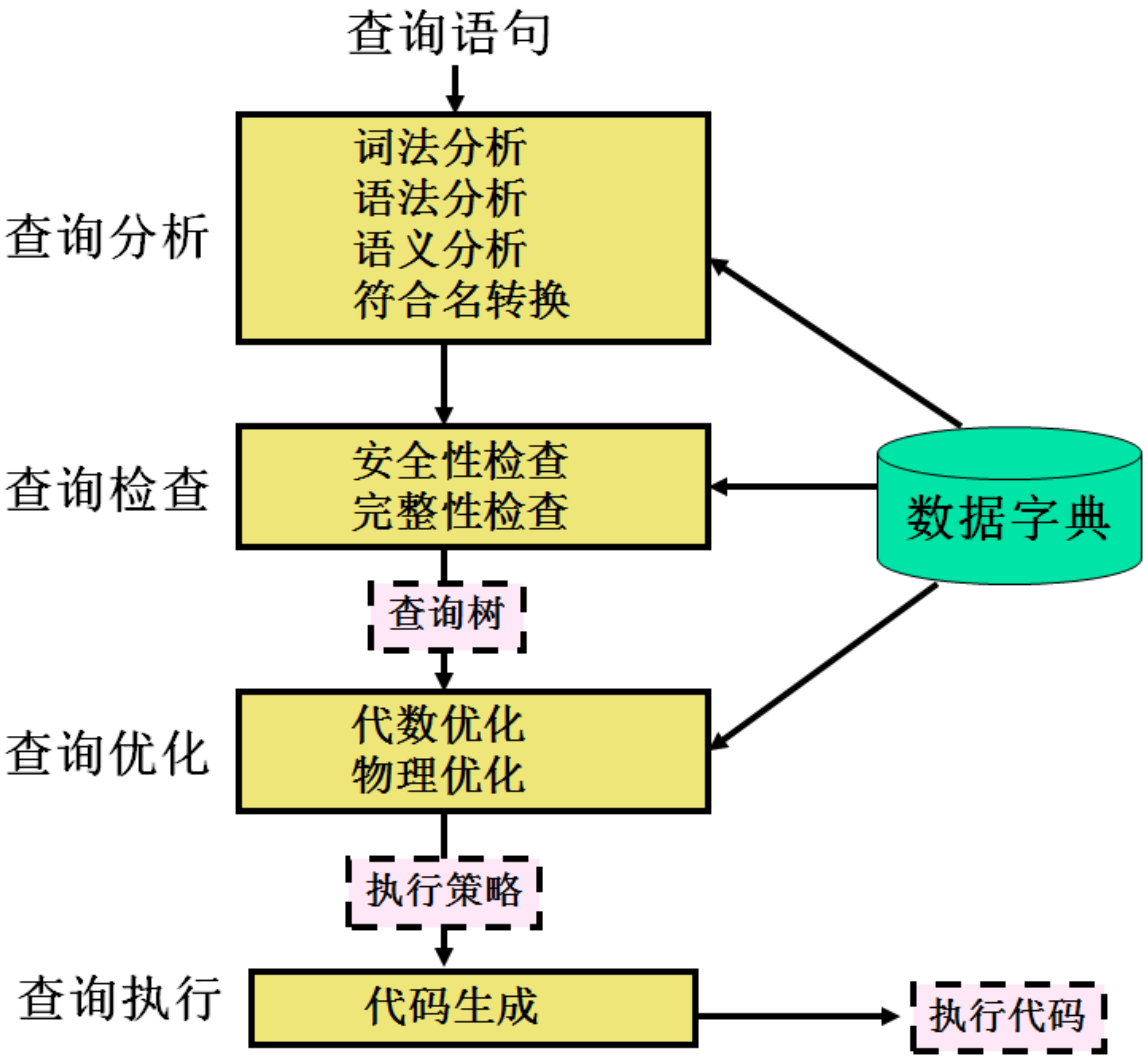
厦门大学计算机科学系

E-mail: ziyulin@xmu.edu.cn

主页：http://www.cs.xmu.edu.cn/linziyu

第9章 数据库查询优化

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

- 查询优化的必要性
  - 查询优化极大地影响**RDBMS**的性能。

- 查询优化的可能性
  - 关系数据语言的级别很高，使**DBMS**可以从关系表达式中分析查询语义。

9.2 关系数据库系统的查询优化

- 用户不必考虑如何最好地表达查询以获得较好的效率
- 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。

(3)优化器可以考虑数百种不同的执行计划，而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

9.2 关系数据库系统的查询优化

查询优化的总目标

选择有效策略，求得给定关系表达式的值

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

查询树

例：有查询Q_1：查询北部地区（AREA='North'）所属部门发出订单的供应商号。

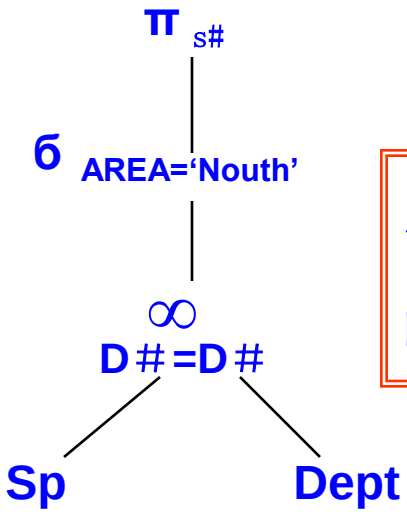
这里涉及两个全局关系 Dept(D#,DNAME,MGTRSSN,AREA) 和 Sp(S#,P#,D#,QUAN)，SQL表达式为：

```
Select S#
From Dept, Sp
Where Sp•D#=Dept.•D# And AREA='North';
```

其相应的代数表达式为：

$$\pi_{S\#}\sigma_{AREA='North'}(Sp \underset{D\#=D\#}{\infty} Dept)$$

其相应的查询树如下：



显然，边为 E_1（$\underset{D\#=D\#}{\infty}$，Sp）时，则Sp是非叶节点 ∞ 的分量。

查询表达式的等价性

[例]：对关系 Emp,有如下SQL查询表达式

```
Select   ENAME,DNO
From     Emp
Where    DNO='15';
```

（4-1）

其相应的代数操作表达式为：

$\pi_{ENAME,\ DNO}(\sigma_{DNO='15'}, Emp)$ （4-2）

或

$\sigma_{DNO='15'}(\pi_{ENAME,\ DNO}\ Emp)$ （4-3）

本例表示了不同的操作顺序仍可获得相同的结果。这就是等价的概念。

[定义4.2]：两个查询表达式 E1 和 E2 是等价的，如果其查询的结果是相同的，记为 E1 ≡ E2。

9.2 关系数据库系统的查询优化

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

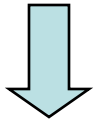
4. 生成查询计划(查询执行方案)

- 查询计划是由一系列内部操作组成的

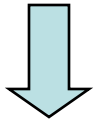
查询树可以理解为全局查询树

根据等价定义，可用三种方式描述查询：

SQL表达式（查询语句）



代数表达式



查询树

代价模型

- 集中式数据库
  - 单用户系统

    总代价 = I/O代价 + CPU代价
  - 多用户系统

    总代价 = I/O代价 + CPU代价 + 内存代价
- 分布式数据库

总代价 = I/O代价 + CPU代价[+ 内存代价] + 通信代价

9.3 基于半联接的查询优化

4.5.1 联接操作重要性
4.5.2 联接操作
4.5.3 半联接操作原理和不对称性
4.5.4 半联接操作的缩减作用
4.5.5 半联接程序的作用
4.5.6 半联接程序的具体过程
4.5.7 半联接技术的不足

9.3.1 联接操作重要性

- 关系数据库由许多关系组成，关系与关系之间的联系主要通过联接操作表现出来，因而在二元操作中，联接操作远比其它操作用得多。
- 讨论联接，其实包括了“选择——投影——联接”的综合问题，即二元操作和一元操作的综合优化问题。
- 分布式查询处理的联接操作，更是影响分布式查询效率的最关键因素。
- 在DDB中，联接操作的大量数据会引起场地间的传输，它直接影响整个系统性能。
- 当前对联接操作的优化有两种趋向：
  - ✓一种是采用半联接技术来减少联接操作的操作数，以降低通讯费用；
  - ✓另一种是直接进行联接操作的代价计算

9.3.2 联接操作

联接操作是从两个关系的笛卡尔积中选取属性间满足一定条件的元组。记作：

$$R \underset{A\,\theta\,B}{\bowtie} S = \{ \widehat{t_r\ t_s} \mid t_r \in R \wedge t_s \in S \wedge t_r[A]\ \theta\ t_s[B] \}$$

其中A和B分别为R和S上可比的属性组。

自然联接（Natural join）是一种特殊的等值联接，它要求两个关系中进行比较的分量必须是相同的属性组，并且要在结果中把重复的属性去掉。即若R和S具有相同的属性组B，则自然连接可记作：

$$R \bowtie S = \{ \widehat{t_r\ t_s} \mid t_r \in R \wedge t_s \in S \wedge t_r[B] = t_s[B] \}$$

等值连接（equi-join），θ为“＝”的连接运算称为等值连接。它是从关系R与S的笛卡尔积中选取A、B属性值相等的那些元组。即等值连接为：

$$R \underset{A=B}{\bowtie} S = \{ \widehat{t_r\ t_s} \mid t_r \in R \wedge t_s \in S \wedge t_r[A] = t_s[B] \}$$

（回顾）自然联接

自然联接的结果是在R和S中的在它们的公共属性名字上相等的所有元组的组合。例如下面是表格“雇员”和“部门”和它们的自然联接：

雇员

Name	EmpId	DeptName
Harry	3415	财务
Sally	2241	销售
George	3401	财务
Harriet	2202	销售

部门

DeptName	Manager
财务	George
销售	Harriet
生产	Charles

雇员 ⋈ 部门

Name	EmpId	DeptName	Manager
Harry	3415	财务	George
Sally	2241	销售	Harriet
George	3401	财务	George
Harriet	2202	销售	Harriet

图 自然联接实例

（回顾）等值联接

θ-联接和等值联接

如果我们要组合来自两个关系的元组，而组合条件不是简单的共享属性上的相等，则有一种更一般形式的连接算子才方便，这就是 θ-联接(或theta-联接)。θ 是在集合{<, ≤, =, >, ≥}中的二元关系。θ的结果由在 R 和 S 中满足关系 θ 的元素的所有组合构成。只有 S 和 R 的表头是不相交的，即不包含公共属性的情况下，θ-连接的结果才是有定义的。

实例：考虑分别列出车模和船模的价格的表“车”和“船”。假设一个顾客要购买一个车模和一个船模，但不想为船花费比车更多的钱。在关系上的θ-联接 *CarPrice* ≥ *BoatPrice* 生成所有可能选项的一个表。

车

CarModel	CarPrice
CarA	20'000
CarB	30'000
CarC	50'000

船

BoatModel	BoatPrice
Boat1	10'000
Boat2	40'000
Boat3	60'000

车 ⋈ 船 $CarPrice \geq BoatPrice$

CarModel	CarPrice	BoatModel	BoatPrice
CarA	20'000	Boat1	10'000
CarB	30'000	Boat1	10'000
CarC	50'000	Boat1	10'000
CarC	50'000	Boat2	40'000

图 θ-联接实例

9.3.3 半联接操作原理和不对称性

半联接操作是关系代数操作中联接（JOIN）操作的一种缩减，关系R和S的半联接记为R∝S。其结果关系是R和S的自然联接（Natural JOIN）后，在R的属性上的投影，可用下述表达式表示：

$R \propto S = \pi_R（R \infty S）$

等价方法：将S中与R有相同属性名的属性集投影出来，然后与R完成自然联接，其等价公式为：

$R \propto S = R \infty （\pi_B S）$

具不对称操作性：$R \propto S \neq S \propto R$。

（回顾）半联接

半联接的结果只是在 S 中有在公共属性名字上相等的元组所有的 R 中的元组。

实例：“雇员”和“部门”和它们的半联接的表：

雇员

Name	EmpId	DeptName
Harry	3415	财务
Sally	2241	销售
George	3401	财务
Harriet	2202	生产

部门

DeptName	Manager
销售	Harriet
生产	Charles

雇员 ⋉ 部门

Name	EmpId	DeptName
Sally	2241	销售
Harriet	2202	生产

图 半联接实例

9.3.4 半联接操作的缩减作用

例4.17 有R（A,B）和 S（B,C），根据半联接计算公式有：$R \underset{B}{\propto} S = R \infty (\pi_B S)$ 和 $S \underset{B}{\propto} R = S \infty (\pi_B R)$ 。如果有图 4.21（a）的实例，则 R $\underset{B}{\propto}$S的结果如 4.21（b）所示，S $\underset{B}{\propto}$ R的结果如图4.21（C）所示。

R（A,B）

A	B
a1	b1
a2	b1
a2	b3
a2	b4
a3	b3

S（B,C）

B	C
b1	c1
b2	c2
b5	c3
b5	c4
b5	c5
b6	c6
b7	c7
b8	c8

(a) 关系R（A,B）和S（B,C）

R′

A	B
a1	b1
a2	b1

(b) $R \underset{B}{\propto} S = R \infty (\pi_B S)$

S′

B	C
b1	c1

(c) $S \underset{B}{\propto} R = S \infty (\pi_B R)$

从图4.21（b）、（c）可得出结论：半联接操作对操作数R或S有缩减作用，且由于其不对称性则各自缩减的程度不一样。半联接操作的缩减性与在联接操作前先对操作数关系做选择和投影有相似的效果。特别在分布式环境中，可用半联接操作减少网上数据的传送量。

图4.12 半联接操作的不对称性与缩减作用

9.3.5 半联接程序的作用

半联接程序是用半联接技术实现联接操作的程序，即用一组具有半联接与联接的操作，映射出具有与等值联接相同结果的过程。

$$R \underset{A=B}{\infty} S \Leftrightarrow (R \underset{A=B}{\propto} \pi_B S) \underset{A=B}{\infty} S \qquad (4\text{-}11a)$$

$$R \underset{A=B}{\infty} S \Leftrightarrow (S \underset{B=A}{\propto} \pi_A R) \underset{B=A}{\infty} R \qquad (4\text{-}11b)$$

（4－11a）、（4－11b）式说明半联接程序与两个关系的直接等值联接等价。

同样，在分布式数据库中，当R，S两个关系不在相同场地上时，用（4－11a）公式或（4－11b）公式处理，可以减少联接操作的数据传送量，并且半联接程序的技术可以缩减它的操作数。

9.3.6 半联接程序的具体过程

以式（4-11a）为例，且假定 R和S不在同一场地：

① 在s场地对S关系做投影操作，使S关系缩减为S′：

$$\pi_B S \Rightarrow S'$$

② 将S′送往r场地；

③ 在r场地上完成R与S′的半联接操作，使R关系缩减为R′：

$$R \underset{A=B}{\propto} S' \Rightarrow R'$$

④ 将R’关系送回S场地；

⑤ 在S场地完成R’与S的联接操作。

$$T = R' \infty S$$

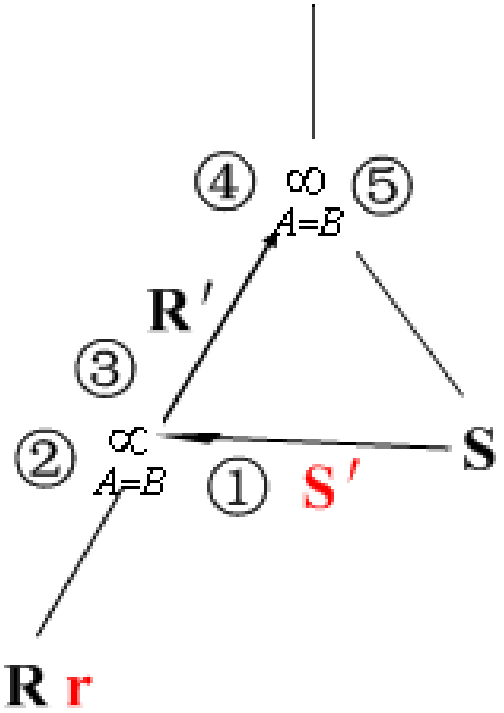


图4.22 半联接程序操作

图4.22的核心技术思想是只将联接操作有关的操作分量在网上进行传输。R与S关系在A=B条件下联接，R、S关系只有公共属性值相等的那些元组才有意义。

9.3.7 半联接技术的不足

- 半联接技术是通过局部缩减操作关系的数据量，发送缩减的关系到执行场地，在执行场地对缩减后的关系进行查询处理。
- 采用该技术大大降低了场地间传递的信息量，从而减少了整个系统的传输代价。
- 但是，半联接技术使传输代价降低的同时，也增加了系统的局部处理代价。因此，半联接技术有如下不足：
  - （1）没有考虑局部代价。
  - （2）当选择度较低时，半联接技术才能够达到减少传输代价的效果。

9.4 基于枚举法的查询优化

- 9.4.1 概述
- 9.4.2 嵌套循环联接算法
- 9.4.3 基于排序的联接算法
- 9.4.4 散列连接算法

9.4.1 概述

- 半联接优化方法能够减少查询执行的通信代价，但是同时会导致通信次数的增加和局部执行代价的增加。
- 当系统环境为高速局域网时，查询执行代价主要考虑的是局部处理代价，半联接优化方法则不再适用。
- 在这种情况下，分布式数据库系统通常使用基于直接联接技术的枚举法优化技术。
- 所谓枚举法优化，就是枚举联接操作所有可行的直接联接算法，通过对每种方法的查询执行代价估计，从中选择一种代价最小的算法作为联接操作的执行算法。
- 直接联接算法广泛应用于集中式数据库系统中，包括：嵌套循环连接算法、归并排序连接算法、散列连接算法和基于索引的连接算法。

9.4.2 嵌套循环联接算法

- 9.4.2.1 基于元组的嵌套循环联接
- 9.4.2.2 基于块的嵌套循环联接
- 9.4.2.3 嵌套循环联接方法的代价估计

9.4.2.1 基于元组的嵌套循环联接

假设有关系R(A,B)和关系S(B,C)，分别有Card(R)=n和Card(S)=m，现在要执行两个关系在属性B上的连接操作，如图4.13所示。

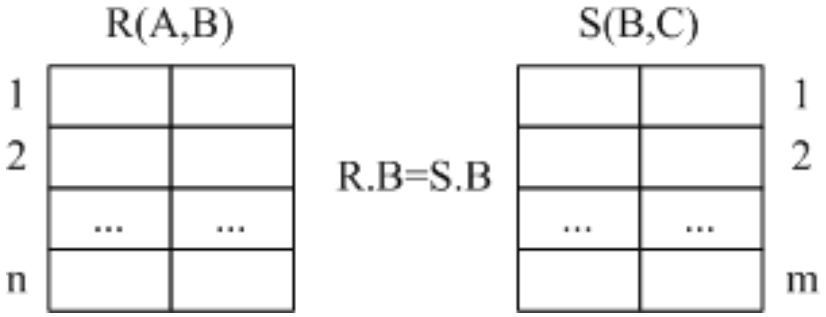


图4.13 两个联接关系

- 嵌套循环连接算法是一种最简单的联接算法，其原理是对联接操作的两个关系对象中的一个仅读取其元组一次，而对另一个关系对象中元组将重复读取。
- 嵌套循环联接算法的特点是可以用于任何大小的关系间的连接操作，不必受连接操作所分配的内存空间大小的限制。
- 对于嵌套循环连接算法，可根据每次操作的对象大小分为基于元组的嵌套循环连接和基于块的嵌套循环连接。

9.4.2.1 基于元组的嵌套循环联接

基于元组的嵌套循环连接

```
Result = {};/*初始化结果集合*/
For each tuple s in S
    For each tuple r in R
        If r.B=s.B then /*元组r和元组s满足连接条件*/
            Join r and  s as tuple t;
            Output t into Result;/*输出连接结果元组*/
        Endif
    Endfor
Endfor
Return Result
```

9.4.2.1 基于元组的嵌套循环联接

- 上面基于元组的嵌套循环连接算法中，对于循环外层的关系，通常称为外关系，而对循环内层的关系称为内关系。
- 在执行嵌套循环连接时，仅对外关系进行1次读取操作，而对内关系则需要进行反复读取操作。
- 如果不进行优化的话，这种基于元组的执行代价很大，以磁盘IO计算最多可能达到Card(R)*Card(S)。
- 因此，通常对这种算法进行修改，以减少嵌套循环连接的磁盘IO代价。一种方法是使用连接属性上的索引，以减少参与连接元组的数量；另一种方法是通过尽可能多地使用内存以减少磁盘IO数目（由此产生了基于块的嵌套循环联接算法）。

9.4.2.2 基于块的嵌套循环联接

- 基于块的嵌套循环连接方法是通过尽可能多地使用内存，减少读取元组所需的I/O次数。其中，对连接操作的两个关系的访问均按块进行组织，同时使用尽可能多的内存来存储嵌套循环中的外关系的块。
- 与基于元组的方法类似，将连接操作中的一个对象作为外关系，每次读取部分元组到内存中，整个关系只读取一次，而另一个对象作为内关系，反复读取到内存中执行连接。
- 对于每个逻辑操作符，数据库系统都会分配一个有限的内存缓冲区。假设为连接操作分配的内存缓冲区大小为M个块，同时有Block(R)≥Block(S) ≥M，即连接的两个关系都不能完全读取到内存中。
- 为此，首先选择较小的关系作为外关系，这里选择关系S。将1到M-1块分配给关系S，而第M块分配给关系R。将外关系S按照M-1个块的大小分为多个子表，并将这些子表依次读取到内存缓冲区中，关系R的每个块会被重复地读入内存和关系S的子表进行连接。
- 对于内存缓冲区中元组的连接操作，先在M-1个块的外关系S元组的连接属性上构建查找结构，再从内关系R在内存中的块中读取元组，通过查找结构与S中的元组连接。

9.4.2.2 基于块的嵌套循环联接

```
Result={};/*初始化结果集合*/
Buffer=M;/*内存缓冲区*/
For each M-1 in Block(S)/*每次从外关系S中读取M-1个块到内存缓冲区中*/
    Read M-1 of Block(S) into Buffer;
    For each block in Block(R) /*每次从内关系R中读取1个块到内存缓冲区*/
        Join M-1 of Block(S)  and 1 of Block(R) in Buffer;/*在内存中对块中元组执行连接*/
        Output t into Result;
    Endfor
Endfor
Return Result;
```

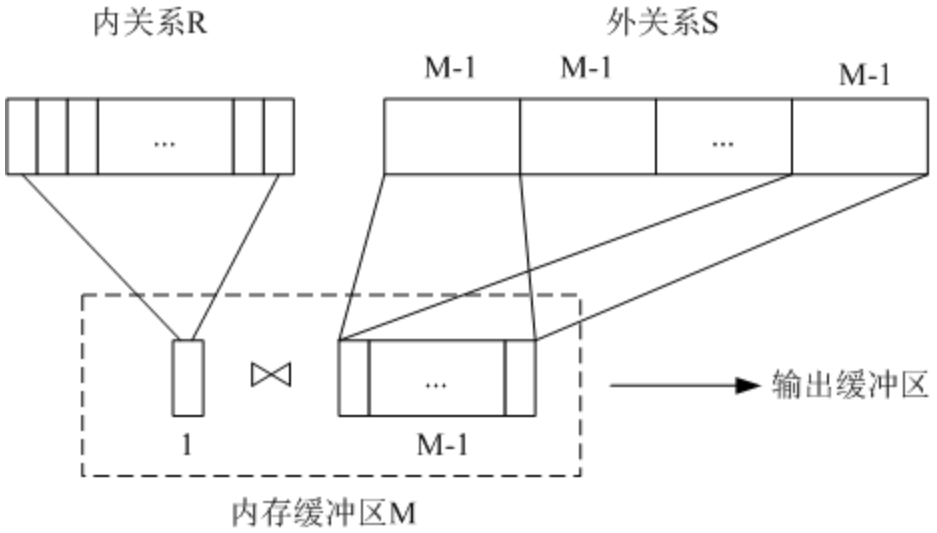


图4.14 基于块的嵌套循环连接算法示意图

9.4.2.3 嵌套循环联接方法的代价估计

- 对于两个关系R和S，如果使用基于元组的嵌套循环连接方法，则需要对每个元组的读取产生1次磁盘IO。因此，假设两个关系的元组数量分别为Card(R)和Card(S)，则基于元组的嵌套循环连接方法的执行代价为Card(R)*Card(S)，即两个关系大小的乘积。
- 如果使用基于块的嵌套循环连接方法，假设两个连接关系R和S占用的块分别为Block(R)和Block(S)，M为内存缓冲区大小。在嵌套过程中使用S作为外关系，每次迭代时首先读取M-1块S的内容到内存缓冲区中，再每次读取R的Block(R)中的1个块的内容到内存中与M-1块的S内容进行连接。因此，连接的代价可以用以下公式计算：

  C_{join}=(Block(S)/(M-1))*(M-1 +Block(R))

  公式可以进一步简化为：

  C_{join}=Block(S) +(Block(S)/(M-1))*Block(R)
- 从上面公式可以看出，在选择较小的关系作为连接的外关系时，可以获得最小的执行代价，因此，通常选择较小的关系作为外关系。

9.4.3 基于排序的联接算法

- 9.4.3.1 概述
- 9.4.3.2 归并排序算法
- 9.4.3.3 简单的基于排序的连接算法
- 9.4.3.4 归并排序连接算法

9.4.3.1 概述

- 基于排序的连接算法，首先将两个关系按照连接属性进行排序，然后按照连接属性的顺序扫描两个关系，同时，对两个关系中的元组执行连接操作。
- 由于数据库中关系的大小往往大于连接操作可用内存缓冲区的大小，因此，对关系的排序通常采用外存排序算法，即归并排序算法。
- 可以将基于排序的连接算法的执行过程与归并排序算法结合的算法，可以节省更多的磁盘IO，通常称为归并排序连接算法。

9.4.3.2 归并排序算法

简单的归并排序算法的执行可以分为两个阶段：

- **第一阶段**：对关系进行分段排序，即首先将需要排序的关系R划分为大小为M个块的子表，其中，M是可用于排序的内存空间的个数，以块为单位，再将每个子表放入内存中采用快速排序等主存排序算法执行排序操作，这样可以获得一组内部已排序的子表。
- **第二阶段**：对关系的子表执行归并操作，即按照顺序从每个排序的子表中读取一个块的内容放入内存，在内存中统一对这些块中的记录执行归并操作，每次选择最大（最小）的记录放入输出缓冲区中，同时删除子表中相应的记录。当子表在内存中的块被取空时，从子表中顺序读取一个新的块放入到内存中继续执行归并操作。

9.4.3.2 归并排序算法

•归并排序的过程如图4.15所示，其中同时对多个子表执行归并操作，因此，也称为两阶段多路归并排序。需要说明的是，第二阶段的归并操作执行的条件是关系的子表数量小于排序操作可用的内存的块数M，这样才能保证同时对所有子表进行归并操作。

•因此，两阶段归并执行的条件是关系的大小Block(R)≤M^2。如果关系的大小大于M^2，则需要嵌套执行归并排序算法，使用三阶段或更多次的归并操作。

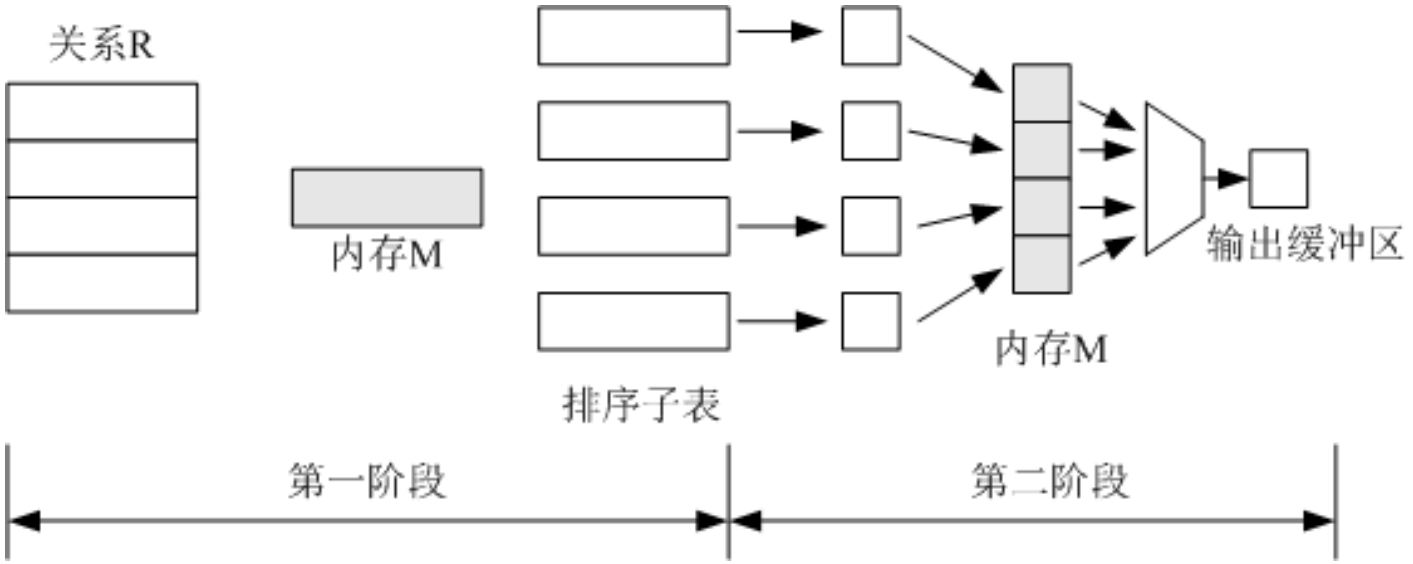


图4.15 简单的归并排序算法的执行

9.4.3.3 简单的基于排序的连接算法

- 基于排序的连接算法，主要是对已经按照连接属性排序的两个关系，按照顺序读取关系中的块到内存中执行连接操作。

代价计算

- 假设在排序阶段使用的是两阶段多路归并排序，关系的大小满足条件Block(R)≤M^2， Block(S) ≤ M^2。这样，算法在排序阶段的执行代价包括对关系的子表执行排序所需的一次读（读子表数据）和一次写（子表排序结果写入磁盘）的代价2(Block(R)+Block(S))，以及多路归并时的读写代价2(Block(R)+Block(S))，而在归并连接阶段还需要对关系执行一次读操作，代价为Block(R)+Block(S)。因此，简单的基于排序的连接算法的执行代价为：C_{join}=5(Block(R)+Block(S))。

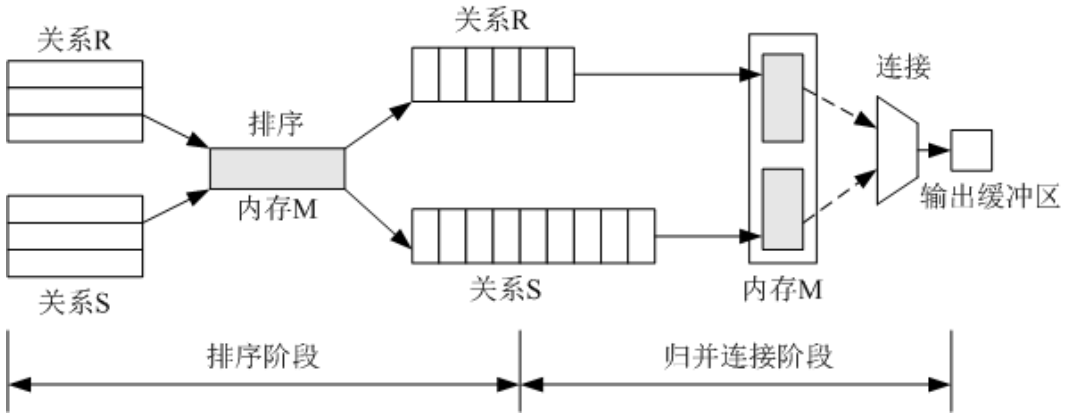


图4.16 简单的排序连接算法

9.4.3.4 归并排序连接算法

- 在简单的基于排序的连接算法中，归并连接阶段仅仅使用了内存缓冲区的两个块的空间，还有大量的空闲内存没有使用。因此，一种更加有效的归并排序连接算法被提出，其思想是将排序的第二阶段与归并连接阶段合并，即直接使用两个关系的排序子表执行归并连接操作，这样可以节省一次对关系的读写操作。
- 假设可用内存缓存区为M个块，算法首先对两个关系划分成大小为M个块的子表并排序，再从每个子表中顺序读取一个块调入内存缓冲区执行连接操作。这里要求两个关系的子表总数不超过M个。

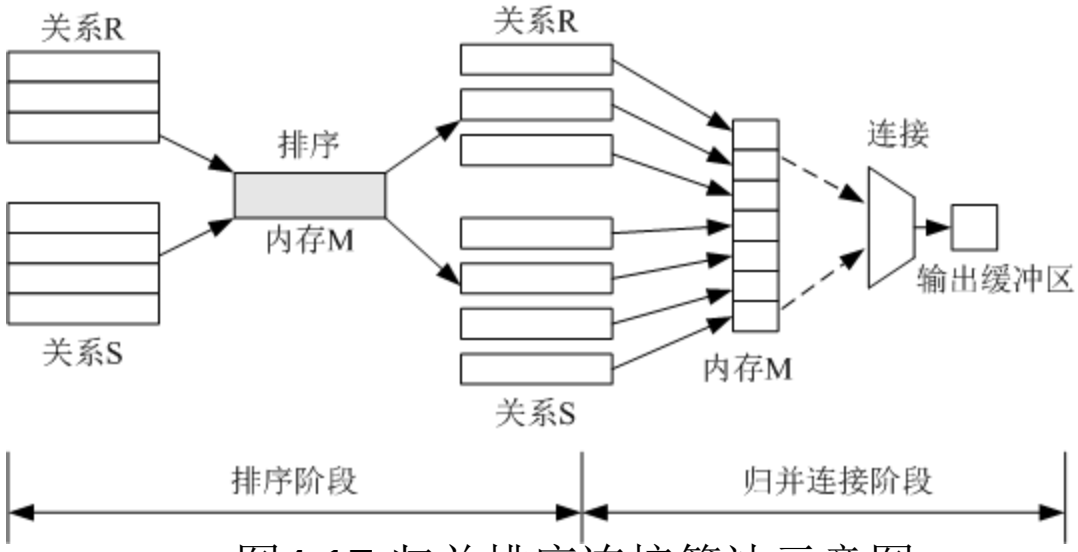


图4.17 归并排序连接算法示意图

9.4.3.4 归并排序连接算法

- 归并排序连接算法在排序阶段的代价包括对子表的一个读写操作2(Block(R)+Block(S))，而在归并连接阶段仅需一次代价为Block(R)+Block(S)的读操作，因此，执行的总代价为：

  C_{join}=3(Block(R)+Block(S))

- 这里注意，归并排序连接算法要求两个关系的子表数量，必须小于内存缓冲区的块数M，这样才能够保证归并阶段有足够的内存存放每个子表的一部分以执行连接。
- 因此，执行归并排序连接算法需要关系的大小满足：

  Block(R)+Block(S) ≤ M^2

9.4.4 散列连接算法

- 散列连接算法，也称为哈希连接算法，基本的执行过程同样分为两个阶段：
  - 第一阶段：使用同一个散列函数，对进行连接的两个关系R和S中的元组的连接属性值进行散列，在连接属性上具有相同键值的元组会出现在相同散列数值的桶中；
  - 第二阶段：对两个关系中散列数值对应的桶中的元组执行连接操作。
- 假设可用的内存缓冲区为M块，散列时使用M-1个块作为桶的缓冲区（最多允许散列到M-1个桶），剩余的1个块作为扫描输入关系的缓冲区。

9.4.4 散列连接算法

- 在算法的第一个阶段中，使用内存将关系R和S散列到M-1个桶中，分别得到写入文件中的$R_1,\ldots,R_{m-1}$和$S_1,\ldots,S_{m-1}$，这个过程需要对两个关系执行一次读写操作，代价为2(Block(R)+Block(S))。

- 在第二个阶段中，每次选取两个关系中具有相同散列值的桶R_i和S_i放到内存中执行连接操作。假设S为较小的关系，由于在对桶连接时必须有一个桶能够全部装入M-1个内存缓冲区块中，才能够保证在执行桶连接时，保证仅执行一次读取操作。因此，关系S的大小需要满足：

  $Block(S) \leq (M-1)^2$

- 若连接的两个关系能够满足一次连接操作的条件，则散列连接算法的执行代价为：

  $C_{join}=3(Block(R)+Block(S))$

附录：主讲教师

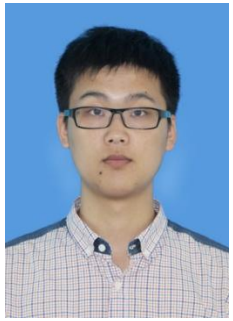
主讲教师：林子雨

单位：厦门大学计算机科学系
E-mail: ziyulin@xmu.edu.cn
个人网页：http://www.cs.xmu.edu.cn/linziyu
数据库实验室网站：http://dblab.xmu.edu.cn

扫一扫访问个人主页

林子雨，男，1978年出生，博士（毕业于北京大学），现为厦门大学计算机科学系助理教授（讲师），曾任厦门大学信息科学与技术学院院长助理、晋江市发展和改革局副局长。中国计算机学会数据库专业委员会委员，中国计算机学会信息系统专业委员会委员，荣获“2016中国大数据创新百人”称号。中国高校首个“数字教师”提出者和建设者，厦门大学数据库实验室负责人，厦门大学云计算与大数据研究中心主要建设者和骨干成员，2013年度厦门大学奖教金获得者。主要研究方向为数据库、数据仓库、数据挖掘、大数据、云计算和物联网，并以第一作者身份在《软件学报》《计算机学报》和《计算机研究与发展》等国家重点期刊以及国际学术会议上发表多篇学术论文。作为项目负责人主持的科研项目包括1项国家自然科学青年基金项目(No.61303004)、1项福建省自然科学青年基金项目(No.2013J05099)和1项中央高校基本科研业务费项目(No.2011121049)，同时，作为课题负责人完成了国家发改委城市信息化重大课题、国家物联网重大应用示范工程区域试点泉州市工作方案、2015泉州市互联网经济调研等课题。中国高校首个“数字教师”提出者和建设者，2009年至今，“数字教师”大平台累计向网络免费发布超过100万字高价值的研究和教学资料，累计网络访问量超过100万次。打造了中国高校大数据教学知名品牌，编著出版了中国高校第一本系统介绍大数据知识的专业教材《大数据技术原理与应用》，并成为京东、当当网等网店畅销书籍；建设了国内高校首个大数据课程公共服务平台，为教师教学和学生学习大数据课程提供全方位、一站式服务，年访问量超过50万次。具有丰富的政府和企业信息化培训经验，厦门大学管理学院EDP中心、浙江大学管理学院EDP中心、厦门大学继续教育学院、泉州市科技培训中心特邀培训讲师，曾给中国移动通信集团公司、福州马尾区政府、福建龙岩卷烟厂、福建省物联网科学研究院、石狮市物流协会、厦门市物流协会、浙江省中小企业家、四川泸州企业家、江苏沛县企业家等开展信息化培训，累计培训人数达3000人以上。

附录：课程助教



助教：魏亮

单位：厦门大学计算机科学系数据库实验室2016级硕士研究生
E-mail: 851189929@qq.com

助教：曾冠华

单位：厦门大学计算机科学系数据库实验室2016级硕士研究生
E-mail: 1393723173@qq.com

附录：班级网站

林子雨主讲《数据库系统原理》2017班级主页

http://dblab.xmu.edu.cn/post/7657/

扫一扫访问班级网站
支持手机浏览


Thank You!