

数据库系统

第9章 查询处理和查询优化 (1)



胡 敏

合肥工业大学

jsjxhumin@hfut.edu.cn

第 9 章 查询处理和查询优化

9.1 查询处理

9.2 查询优化

9.3 代数优化

9.4 物理优化

9.5 小结



第9章 查询处理和查询优化

■ 基本内容

➤ 查询处理

- ✓ 查询处理步骤
- ✓ 查询分析
- ✓ 查询检查
- ✓ 查询优化
- ✓ 查询执行

➤ 查询优化

- ✓ 为什么要查询优化?
- ✓ 查询优化的基本思路
- ✓ 逻辑查询优化
- ✓ 物理查询优化



第9章 查询处理和查询优化

■ 重点与难点

理解查询优化的整体思路是什么？

理解并掌握基于关系代数进行逻辑查询优化的方法和原则

理解物理查询优化中的代价估算方法



查询处理和查询优化

- 如何以有效的方式处理用户查询是RDBMS有效实现的关键问题之一
- 查询处理的中心任务是使用诸如SQL这样的说明性语言表达的用户查询转换成一序列能够在物理文件上执行的操作，并执行这些操作得到查询结果。
- 查询优化是查询处理的关键步骤，它从众多的查询执行方案中选择最有效的执行方案。



查询处理和查询优化

■ 为什么需要查询优化

➤ 关系数据库的执行效率问题

一个例子: $\Pi_{S\ name}(\sigma_{Student.Sno=SC.Sno \wedge Course.no=SC.no \wedge SC.Cno='2'}(Student \times SC \times Course))$

Student: 10000个学生记录(每年2500, 四年)

Course: 1000门课程记录

SC: 10000*50条选课记录(注: 10000学生, 每人选50门课程)

Student \times SC \times Course

10000*10000*50*1000条记录=5*10¹²条记录



查询处理和查询优化

■ 为什么需要查询优化

➤ 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？

➤ $\Pi_{Sno, Sname, Score}(\sigma_{Course.Cno='2'}((Student \bowtie SC) \bowtie Course))$

10000	500000
500000	1000
500000	

$\Pi_{Sno, Sname, Score}((Student \bowtie (SC \bowtie (\sigma_{Course.Cno='2'}(Course))))$

1000
500000
1
2500(约)
2500(约)

关系代数的操作
顺序重要吗？



查询处理和查询优化

■ 关系查询优化是影响RDBMS性能的关键因素

★“如何使数据库查询的执行时间最短？”

★三个层面进行优化：

✓ 语义优化：利用模型的语义及完整性规则，优化查询。（用户/程序员优化？）

关系代数的有效的等价表达式； 不同表达形式与查询效率间的必然联系

• 尽可能早做选择运算
• 尽可能早做投影运算
• 改写成等价的效果更好的语句

DBMS自动完成？

✓ 语法优化---逻辑层优化：利用语法结构，优化操作执行顺序；

✓ 执行优化---物理层优化：存取路径和执行算法的选择与执行次序优化；

获取较高查询效率表达式的算法。

基于不同算法的实现
程序构造

DBMS自动完成？



第 9 章 查询处理和查询优化

9.1 查询处理

9.2 查询优化

9.3 代数优化

9.4 物理优化

9.5 小结



9.1 查询处理

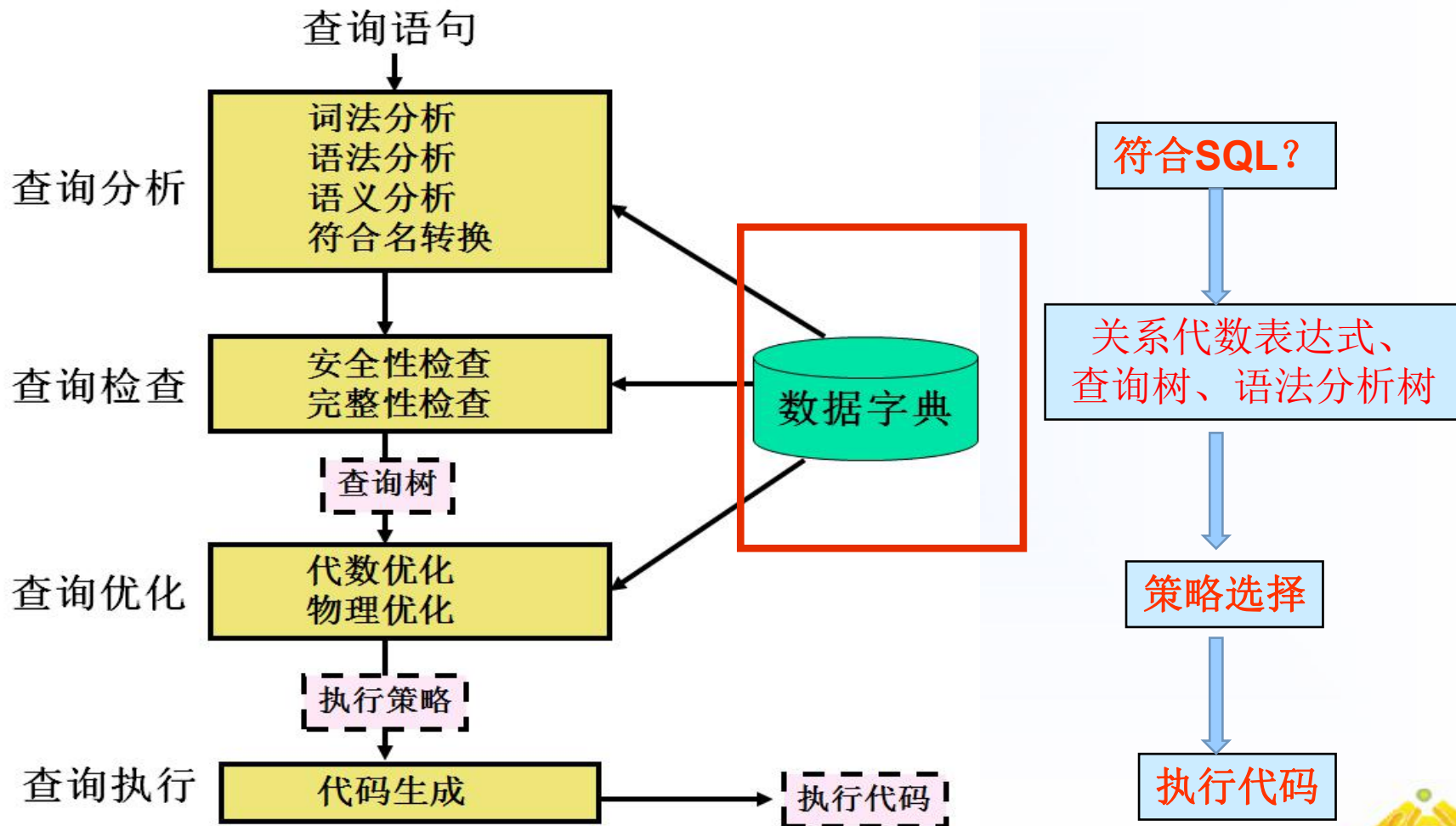
9.1.1 查询处理步骤

9.1.2 实现查询操作的算法示例



9.1 查询处理

查询处理——从查询语句出发，到获得查询结果的处理过程。



RDBMS 数据字典

■ SQL Server

➤ 例如：列出数据库的表结构

```
SELECT T.name AS 表名, C.name AS 列名,  
       S.name AS 类型, C.length 长度,  
       C.isnullable 可否空,  
       C.colorder 列序, P.value AS 含义  
FROM   dbo.sysobjects AS T  
INNER JOIN  dbo.syscolumns AS C ON T.id = C.id  
LEFT OUTER JOIN sys.extended_properties AS P  
    ON C.id = P.major_id AND C.colid = P.minor_id  
INNER JOIN  dbo.systypes AS S ON C.xtype = S.xtype  
WHERE (T.xtype = 'U') and  
       T.name in ('student', 'course', 'sc')  
ORDER BY 表名, C.colid
```

表名	列名	类型	长度	可否空	列序	含义
course	cno	char	4	0	1	课程号
course	cname	char	40	0	2	课程名
course	cpno	char	4	1	3	先行课号
course	ccredit	int	4	1	4	学分
SC	sno	char	9	0	1	学号
SC	cno	char	4	0	2	课程号
SC	grade	smallint	2	1	3	成绩
student	sno	char	9	0	1	学号
student	sname	char	20	1	2	姓名
student	ssex	char	2	1	3	性别
student	sage	smallint	2	1	4	年龄
student	sdept	char	20	1	5	系别



RDBMS 数据字典

■ MySQL

➤ 例如：列出数据库（网上商城）的表结构

```
SELECT a.TABLE_NAME as 表名,  
       a.COLUMN_NAME as 列名,  
       a.ORDINAL_POSITION 列序号,  
       a.IS_NULLABLE as 是否允许空,  
       a.COLUMN_TYPE as 类型,  
       a.COLUMN_COMMENT as 说明  
From information_schema.COLUMNS  
LEFT JOIN information_schema.TABLES  
       a.TABLE_NAME=b.TABLE_NAME  
       a.TABLE_SCHEMA = b.TABLE_SCHEMA  
where a.TABLE_SCHEMA='shop'  
      b.TABLE_TYPE = 'BASE TABLE'  
      a.table_name like 'orders'  
ORDER BY b.TABLE_NAME
```

给数据对象写注释是个好习惯！

名	类型	长度	小数点	不是 null	虚拟	键	注释
ordersID	char	12	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	1	订单号
ordersDate	date	0	0	<input type="checkbox"/>	<input type="checkbox"/>		日期
custID	char	8	0	<input type="checkbox"/>	<input type="checkbox"/>		客户号
shipTo	int	11	0	<input type="checkbox"/>	<input type="checkbox"/>		邮寄地址
transactor	char	10	0	<input type="checkbox"/>	<input type="checkbox"/>		经办人
approvedBy	char	10	0	<input type="checkbox"/>	<input type="checkbox"/>		审核人
status	char	1	0	<input type="checkbox"/>	<input type="checkbox"/>		状态
operator	char	10	0	<input type="checkbox"/>	<input type="checkbox"/>		操作员

表名	列名	列序号	是否允许空	类型	说明
orders	ordersID	1	NO	char(12)	订单号
orders	ordersDate	2	YES	date	日期
orders	custID	3	YES	char(8)	客户号
		4	YES	int(11)	邮寄地址
		5	YES	char(10)	经办人
		6	YES	char(10)	审核人
		7	YES	char(1)	状态
		8	YES	char(10)	操作员
		1	NO	int(11)	流水号
		2	NO	char(12)	订单号
		3	YES	int(11)	行号
		4	NO	char(10)	商品号
		5	NO	int(11)	数量
		6	YES	decimal(8,2)	单价



RDBMS 数据字典

■ SQL Server

数据对象的外部名称和内部名称

```
select * from sys.objects where name in ('student', 'sc', 'course')
```

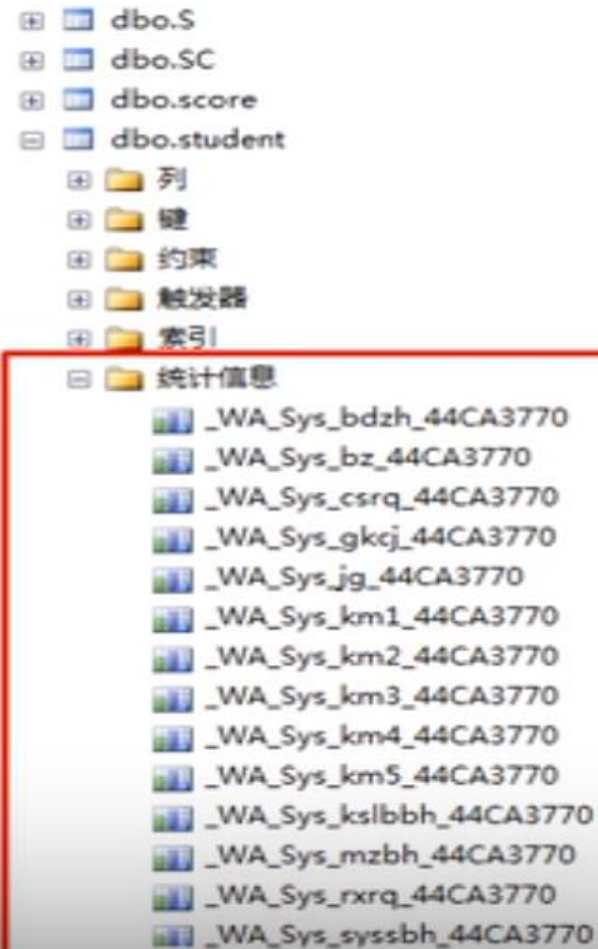
	name	object_id	principal_id	schema_id	parent_object_id	type	type_desc	create_date
1	course	277576027	NULL	1	0	U	USER_TABLE	2018-04-11
2	SC	309576141	NULL	1	0	U	USER_TABLE	2018-04-11
3	student	245575913	NULL	1	0	U	USER_TABLE	2018-04-11
4	student	373576369	NULL	5	0	U	USER_TABLE	2018-04-11



RDBMS 数据字典

■ SQL Server

➤ 统计信息

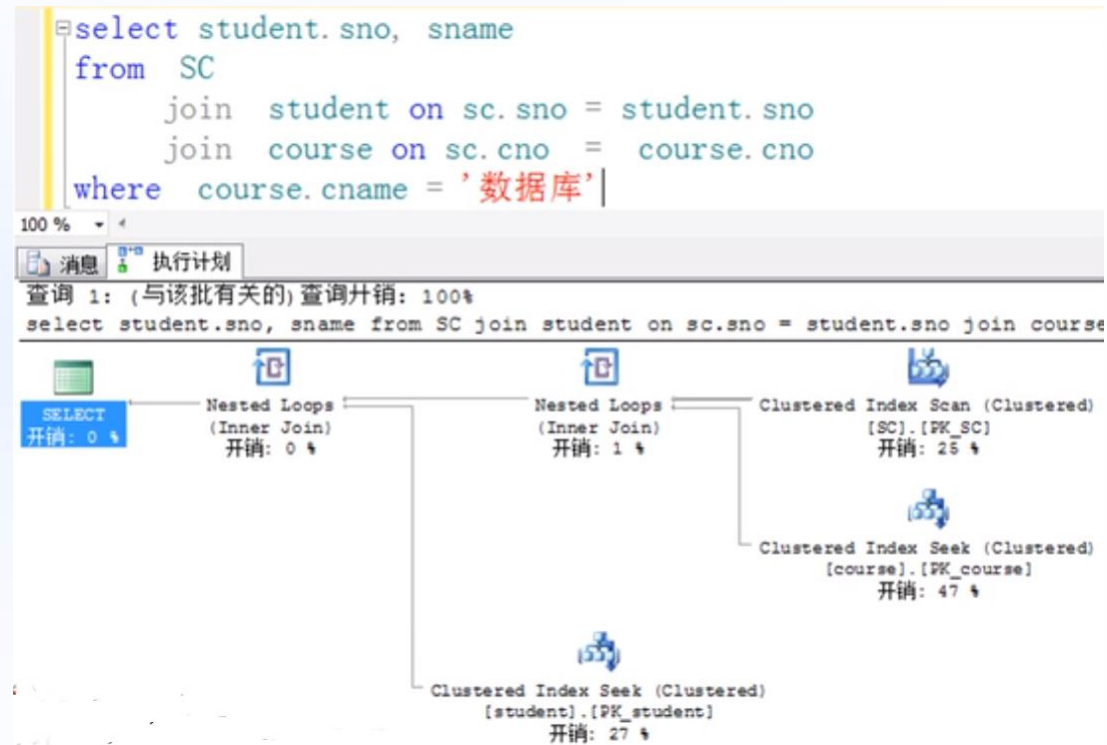
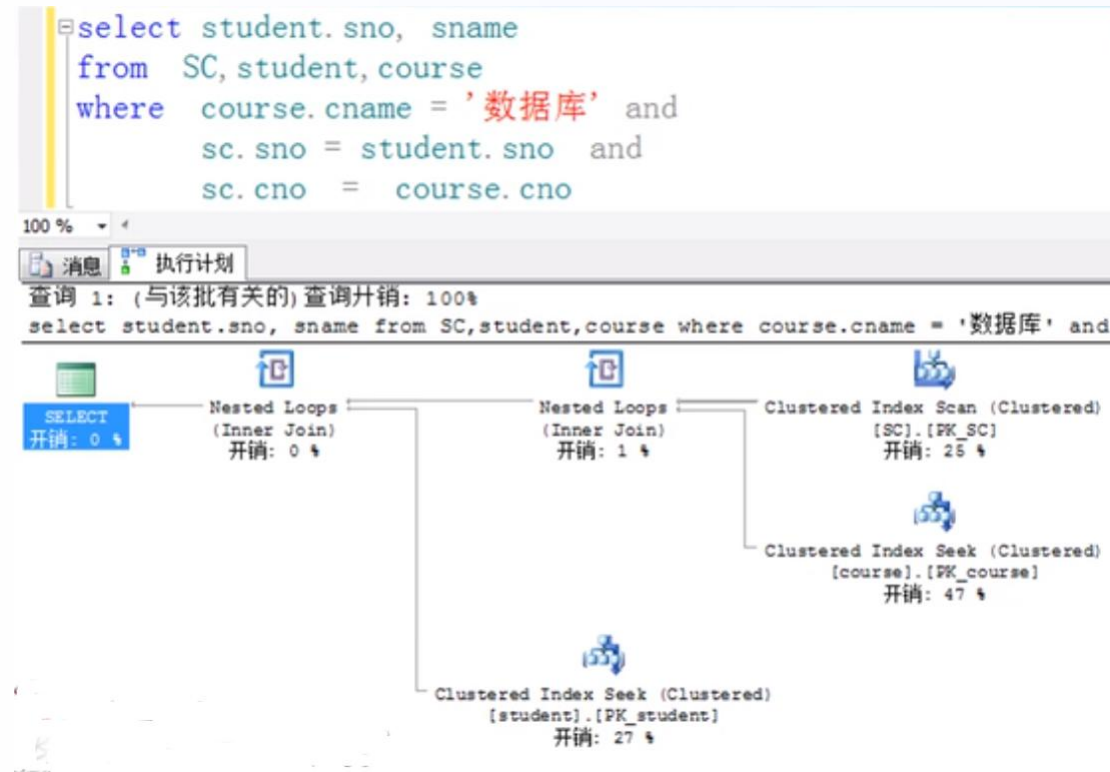


- dbo.S
- dbo.SC
- dbo.score
- dbo.student
 - 列
 - 键
 - 约束
 - 触发器
 - 索引
 - 统计信息
 - _WA_Sys_bdzh_44CA3770
 - _WA_Sys_bz_44CA3770
 - _WA_Sys_csrq_44CA3770
 - _WA_Sys_gkcj_44CA3770
 - _WA_Sys_jg_44CA3770
 - _WA_Sys_km1_44CA3770
 - _WA_Sys_km2_44CA3770
 - _WA_Sys_km3_44CA3770
 - _WA_Sys_km4_44CA3770
 - _WA_Sys_km5_44CA3770
 - _WA_Sys_kslbbh_44CA3770
 - _WA_Sys_mzbh_44CA3770
 - _WA_Sys_rxrq_44CA3770
 - _WA_Sys_syssbh_44CA3770



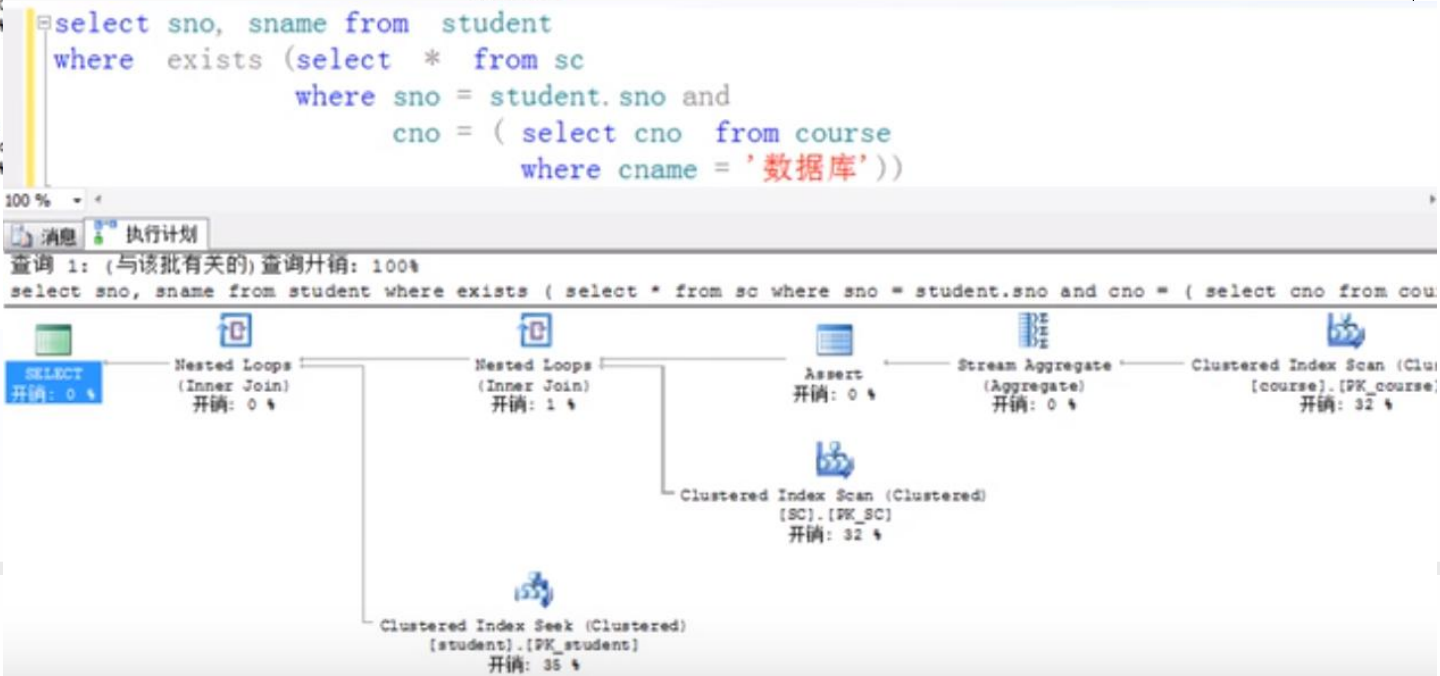
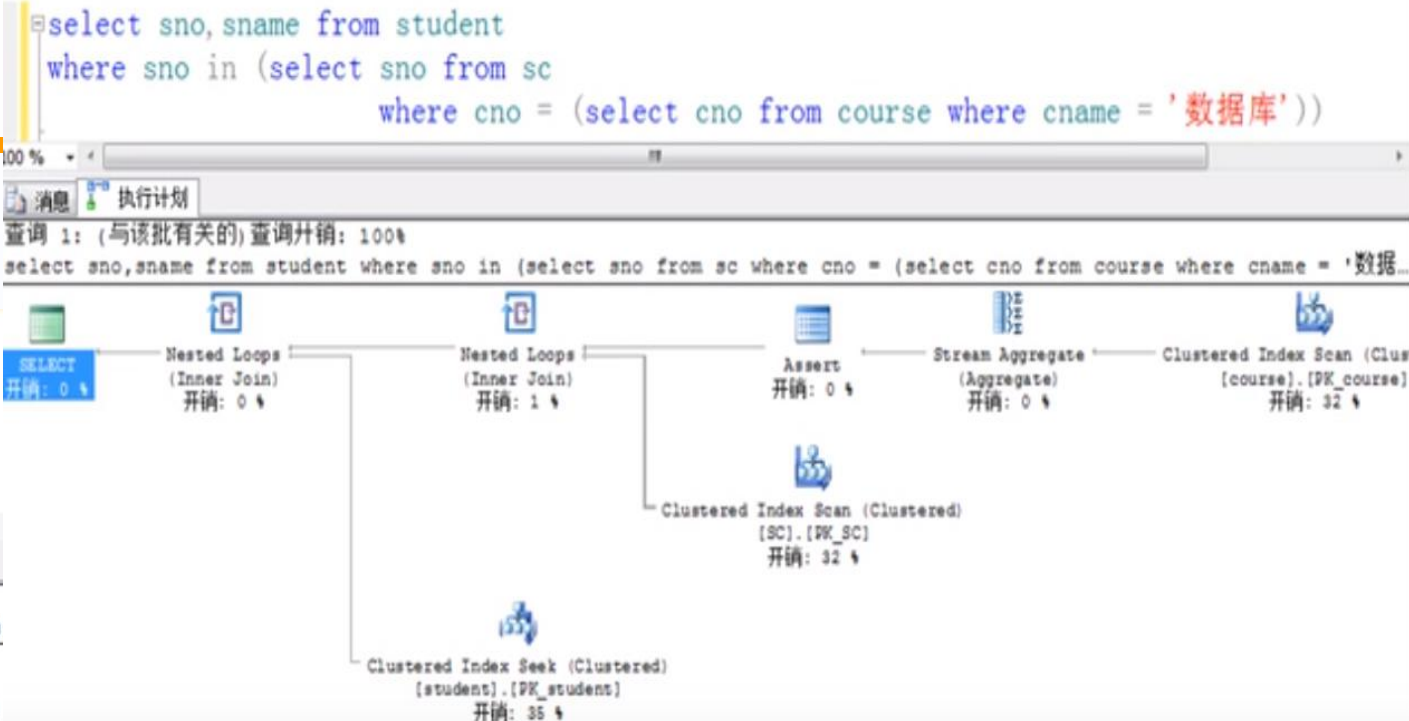
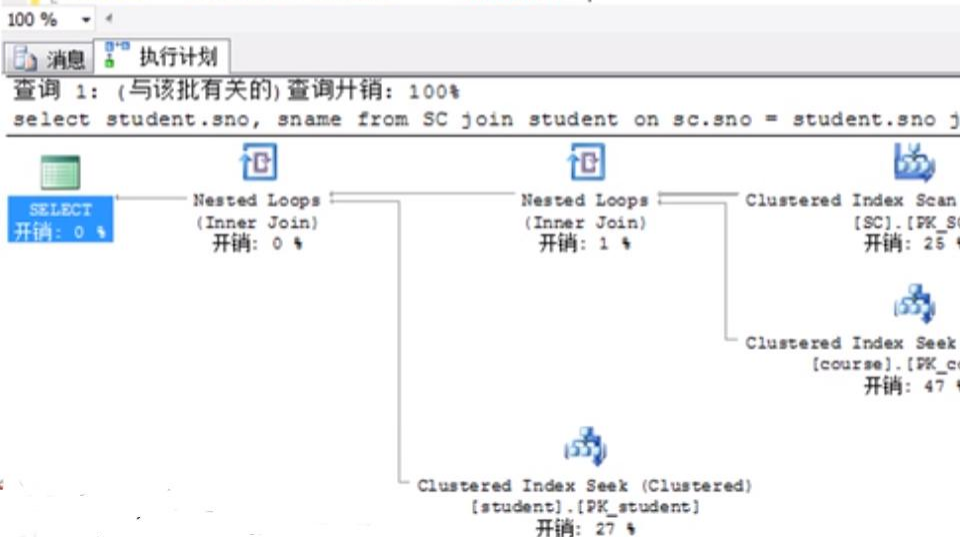
RDBMS 数据字典

■ SQL Server



■ SQL Server

```
select student.sno, sname
from SC
    join student on sc.sno = student.sno
    join course on sc.cno = course.cno
where course.cname = '数据库'
```



RDBMS 数据字典

■ MySQL

```
1 explain select student.sno, sname
2 from SC, student, course
3 where course.cname = '数据库' and
4       sc.sno = student.sno and
5       sc.cno = course.cno |
```

信息	Result 1	剖析	状态
Status	Duration	Percentage	
▶ starting	0.000049	4.584	
checking permissions	0.00001	0.935	
Opening tables	0.000018	1.684	
init	0.000043	4.022	
System lock	0.000008	0.748	
optimizing	0.000003	0.281	
optimizing	0.000002	0.187	
statistics	0.000012	1.123	



9.1.1 查询处理步骤 (1)

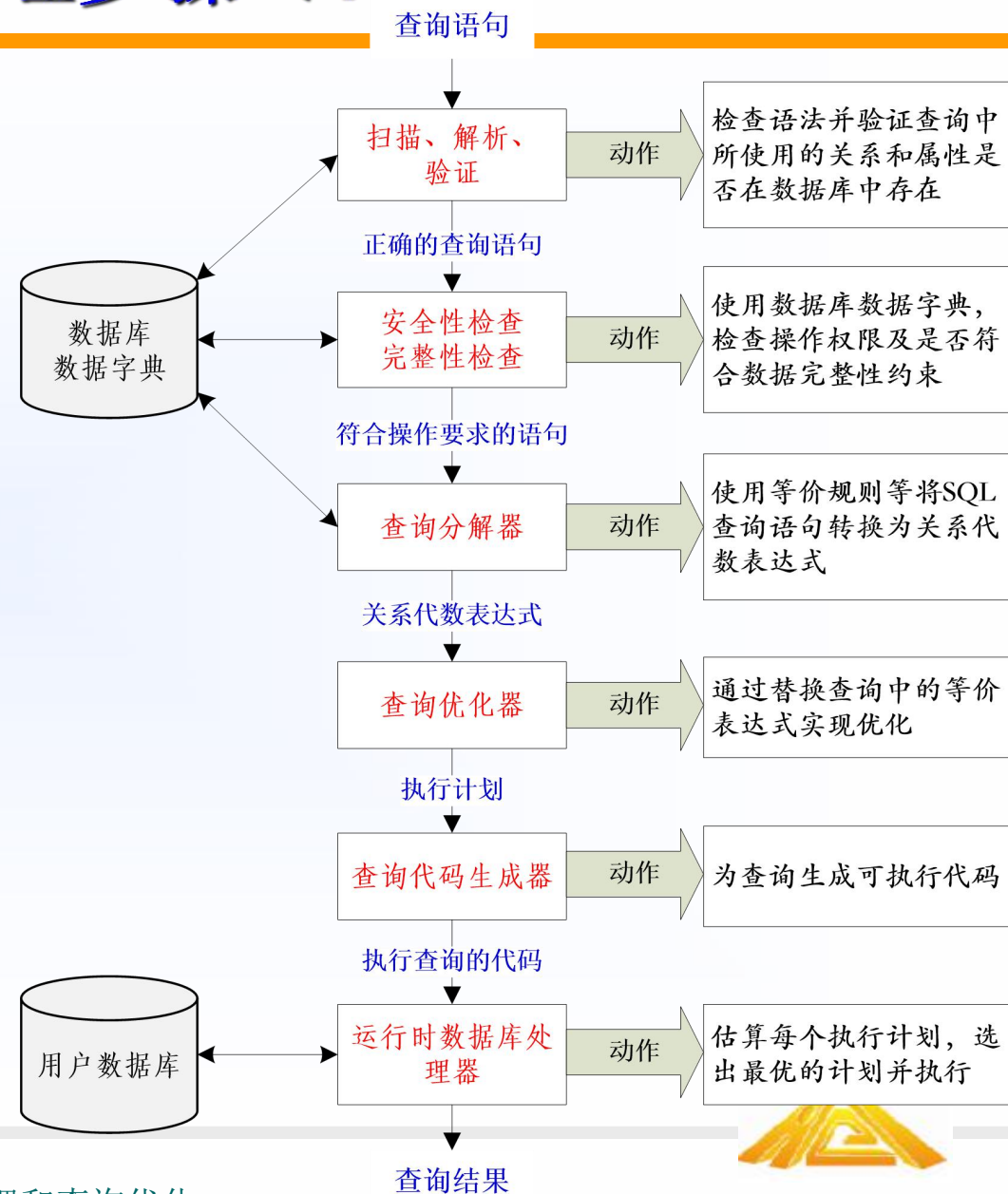
■ 1. 查询处理的任务

把用户提交给RDBMS的查询语句转换为高效的执行计划。

■ 2. 基本步骤 Basic Steps in Query Processing

- 1. 查询分析
- 2. 查询检查
- 3. 查询优化
- 4. 查询执行

查询处理的典型过程



9.1 查询处理

9.1.1 查询处理步骤

9.1.2 实现查询操作的算法示例



9.1.2 实现查询操作的算法示例

- 一、选择操作的实现
- 二、连接操作的实现



一、选择操作的实现

[例1] `Select * from student where <条件表达式>` ;
考虑<条件表达式>的几种情况:

C1: 无条件;

C2: `Sno='200215121'`;

C3: `Sage>20`;

C4: `Sdept='CS' AND Sage>20`;



选择操作典型实现方法：

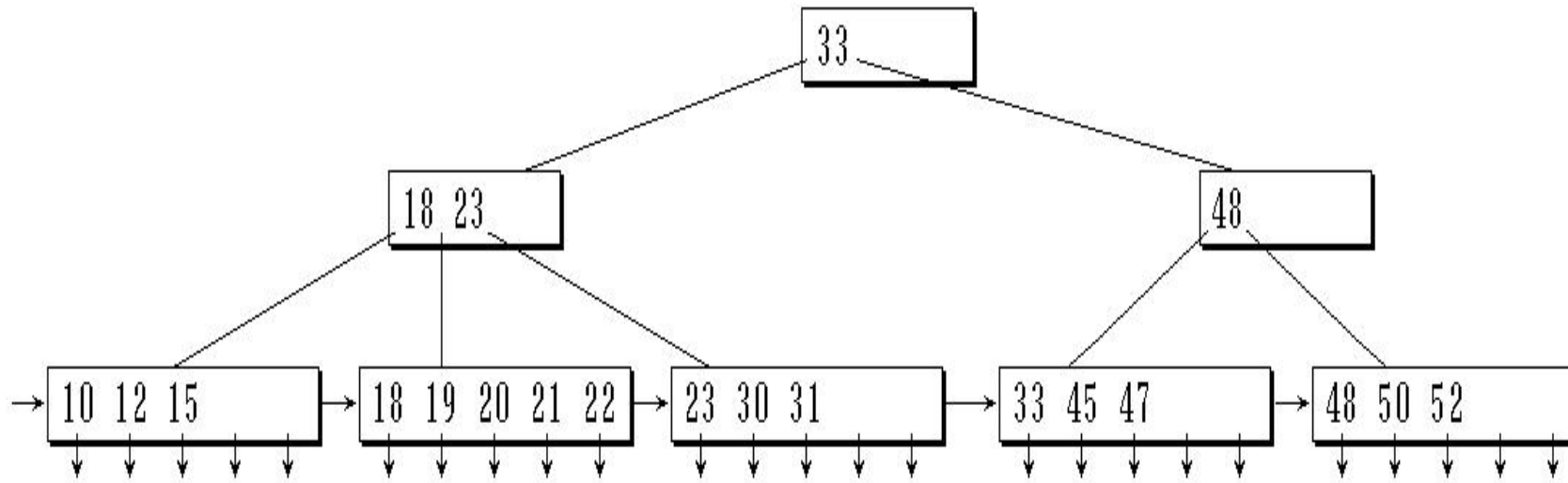
1. 简单的全表扫描方法

- 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
- 适合小表，不适合大表

2. 索引(或散列)扫描方法

- 适合选择条件中的属性上有索引(例如B+树索引或Hash索引)
- 通过索引先找到满足条件的元组主码或元组指针，再通过元组指针直接在查询的基本表中找到元组





- 散列方法是在表项的存储位置与它的关键码之间建立一个确定的对应函数关系 $Hash()$ ，使每个关键码与结构中一个唯一存储位置相对应：
 $Address = Hash(Rec.key)$
- 在存放表项时，依相同函数计算存储位置，并按此位置存放。这种方法就是散列方法。在散列方法中使用的转换函数叫做散列函数。而按此种想法构造出来的表或结构就叫做散列表。
- 使用散列方法进行搜索不必进行多次关键码的比较，搜索速度比较快，可以直接到达或逼近具有此关键码的表项的实际存放地址。



选择操作的实现（续）

[例1-C1] 无条件 --全表扫描

[例1-C2] 以C2为例，Sno= ‘200215121’，并且Sno上有索引(或Sno是散列码)

- 使用索引(或散列)得到Sno为 ‘200215121’ 元组的指针
- 通过元组指针在student表中检索到该学生

[例1-C3] 以C3为例，Sage>20，并且Sage 上有B+树索引

- 使用B+树索引找到Sage=20的索引项，以此为入口点在B+树的顺序集上得到Sage>20的所有元组指针
- 通过这些元组指针到student表中检索到所有年龄大于20的学生。



选择操作的实现（续）

[例1-C4] 以C4为例，**Sdept= 'CS' AND Sage>20**，如果**Sdept**和**Sage**上都有索引：

算法一：分别用上面两种方法分别找到**Sdept= 'CS'**的一组元组指针和**Sage>20**的另一组元组指针

- 求这**2**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'**的一组元组指针，

- 通过这些元组指针到**student**表中检索
- 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。



二、连接操作的实现

连接操作是查询操作中最耗时的操作之一

[例2] SELECT * FROM Student, SC
 WHERE Student.Sno=SC.Sno;

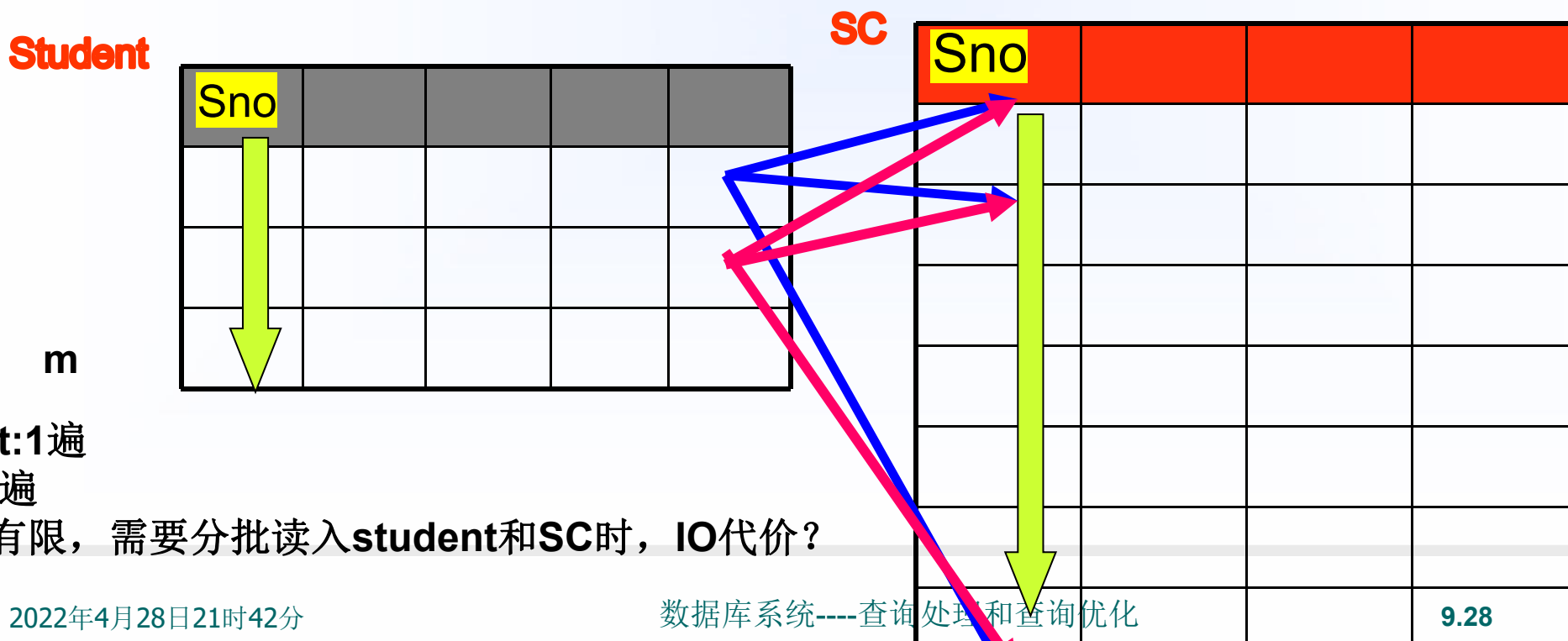
- 1. 嵌套循环方法(nested loop)
- 2. 排序-合并方法(sort-merge join 或merge join)
- 3. 索引连接(index join)方法
- 4. Hash Join方法



连接操作的实现（续）

1. 嵌套循环方法(nested loop)

- 对外层循环(Student)的每一个元组(s)，检索内层循环(SC)中的每一个元组(sc)
- 检查这两个元组在连接属性(sno)上是否相等
- 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



student: 1遍

SC: m遍

当内存有限，需要分批读入student和SC时，IO代价？



连接操作的实现（续）

2. 排序-合并方法(sort-merge join 或merge join)

➤ 适合连接的诸表已经排好序的情况

➤ 排序—合并连接方法的步骤:

①如果连接的表没有排好序，先对**Student**表和**SC**表按连接属性**Sno**排序

②取**Student**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**Student**表扫描它的下一个元组，再扫描**SC**表中具有相同**Sno**的元组，将它们连接起来

④重复上述步骤直到**Student**表扫描完

student:扫描1遍
SC: 扫描1遍
排序时间



排序-合并连接方法示意图



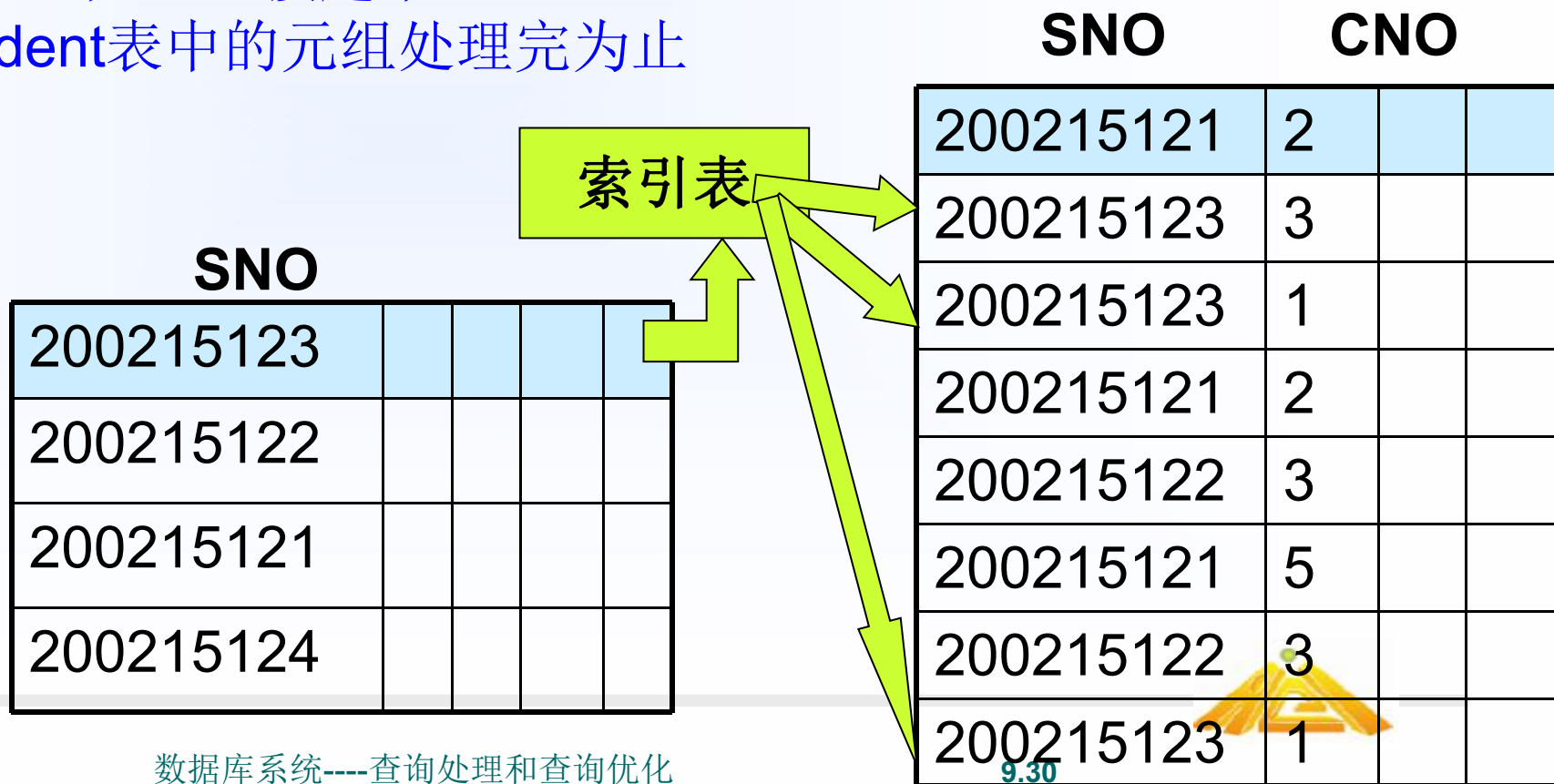
连接操作的实现（续）

3. 索引连接(index join)方法

■ 步骤:

- ① 在SC表上建立属性Sno的索引，如果原来没有该索引
 - ② 对Student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Student元组连接起来
- 循环执行②③，直到Student表中的元组处理完为止

两边都有索引=排序合并
一边有索引：扫描一遍另一表

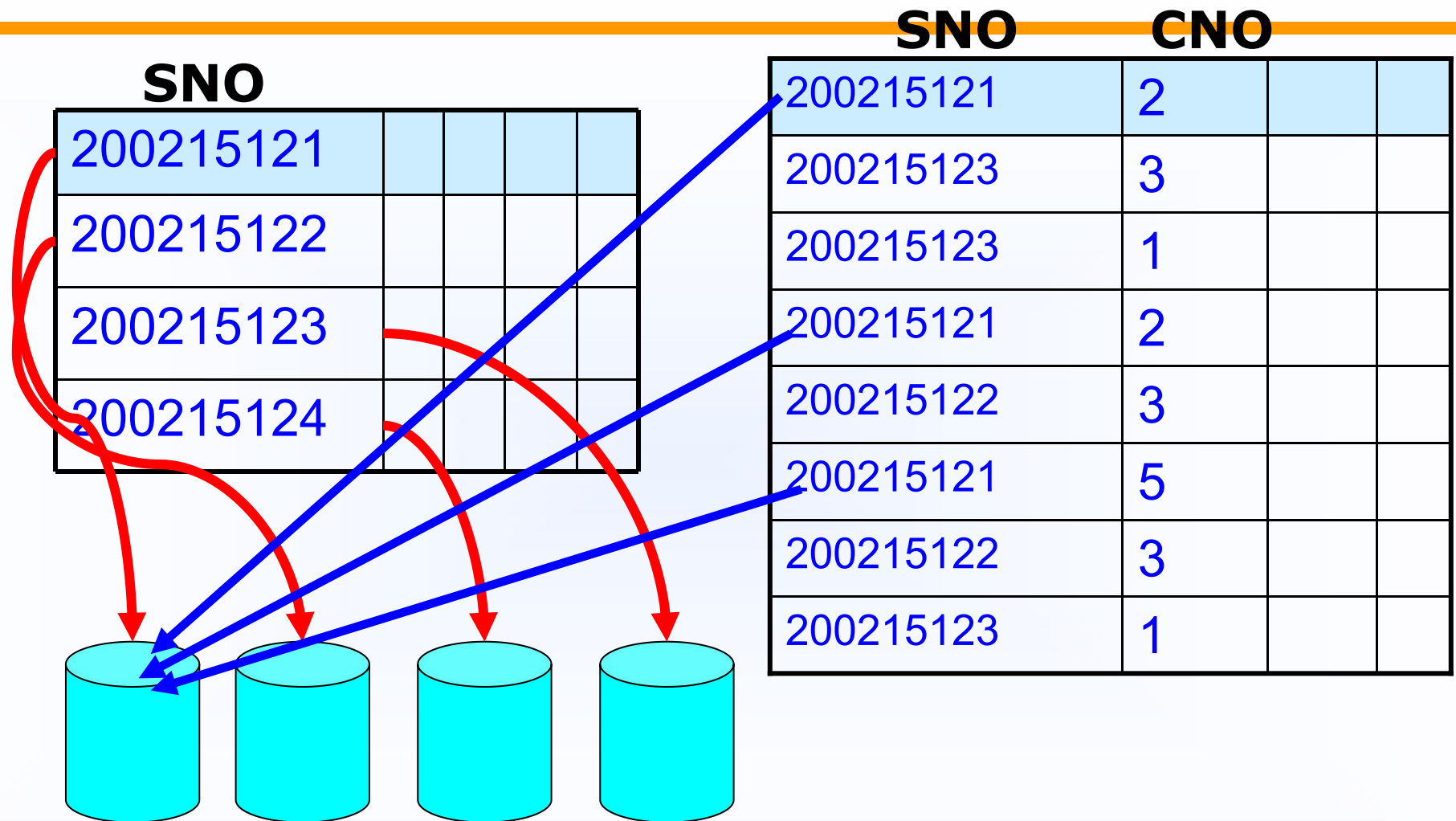


连接操作的实现（续）

4. Hash Join方法

- 把连接属性作为hash码，用同一个hash函数把R和S中的元组散列到同一个hash文件中
- 步骤：
 - ✓ 划分阶段(partitioning phase):
 - ★ 对包含较少元组的表(比如R)进行一遍处理
 - ★ 把它的元组按hash函数分散到hash表的桶中
 - ✓ 试探阶段(probing phase): 也称为连接阶段(join phase)
 - ★ 对另一个表(S)进行一遍处理
 - ★ 把S的元组散列到适当的hash桶中
 - ★ 把元组与桶中所有来自R并与之相匹配的元组连接起来





- **hash join**算法前提：假设两个表中较小的表在第一阶段后可以完全放入内存的**hash**桶中
- 以上的算法思想可以推广到更加一般的多个表的连接算法上



下课了。。



休息一会儿。。。。

