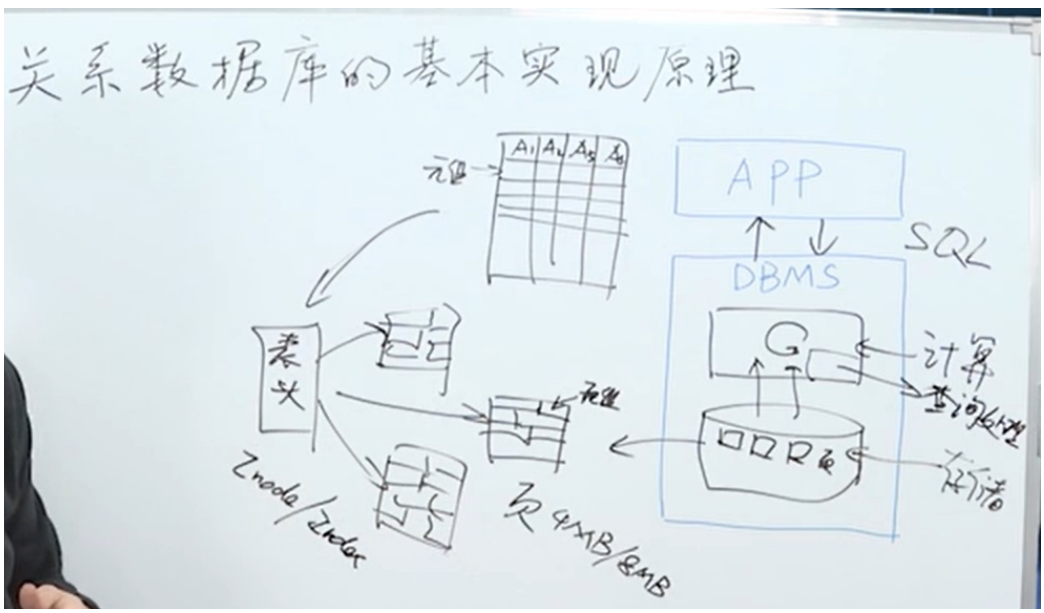
6.1 关系数据库的基本架构

从存储层获取数据到计算层，然后在计算层进行计算，得到sql语句的查询结果，将结果返回给应用。

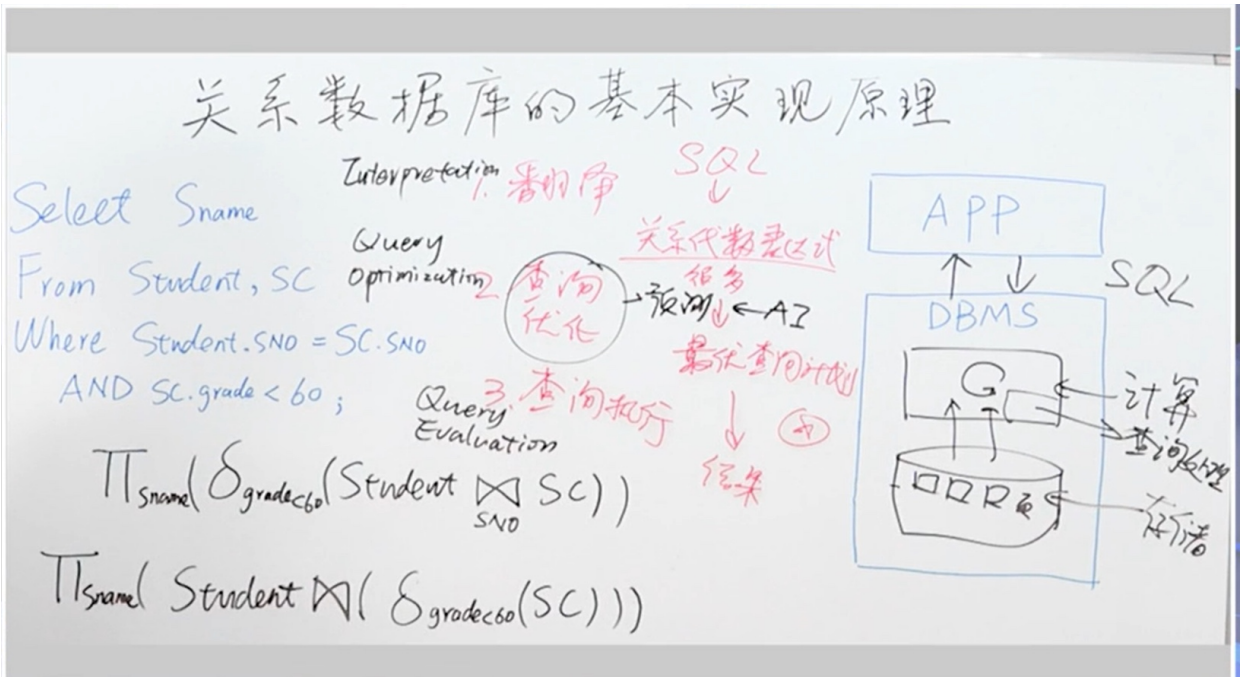


6.2 sql查询的执行过程

1 翻译 sql 翻译成关系代数表达式（不唯一）

2 查询优化 找到最优的查询计划

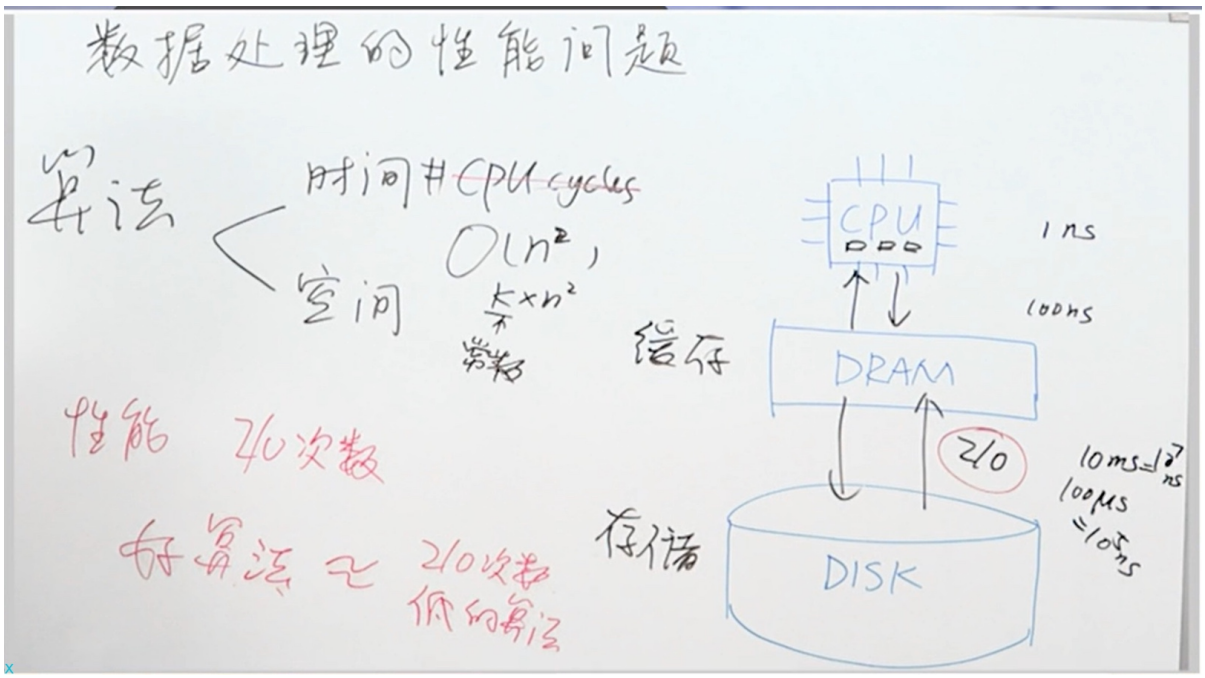
3 执行查询



6.3 数据处理的性能问题

对于一个数据密集型的问题而言，cpu常常是很空闲的，你需要等数据从磁盘被加载到内存，再从内存到cpu的寄存器

如果用Cpu的计算次数来衡量一个数据处理程序的执行时间，其实没有考虑到cpu本身计算的速度（很快）和它访问内存速度（很慢）之间的差异。如果算法是数据密集型的，有可能对于cpu而言，它大部分时间是在访问内存的数据，而不是在寄存器里面做计算。所以最后影响算法执行效率的并不是cpu cycles而是CPU访问内存的次数。当设计一个数据处理算法的时候，要优化的东西是I/o的次数



6.4 选择算子的实现

选择是最简单的计算算子

索引 找到拥有某个键值key的元组所在的位置，可以快速定位到你想要的元组

扫描 假设有一个inside 结构，，得到包含所有数据的页面，一个个页面依次访问，遍历表里面的数据，比较耗时

索引访问/索引扫描 通过索引访问被你选择的元组，条件是选择的操作使用的选择的条件一定是包含索引的键值的，只需要访问若干个页面就可以找到你想要的数据，不需要全部遍历。效率往往比扫描高。

但并不是所有情况下都更适合使用索引，一旦索引的key值经常被更新，你就需要经常改动并且维护它，这时候不见得使用索引更好。还有些时候，使用索引访问数据起不到提升效率的作用（特殊情况，满足要求的数据项很多很广泛）

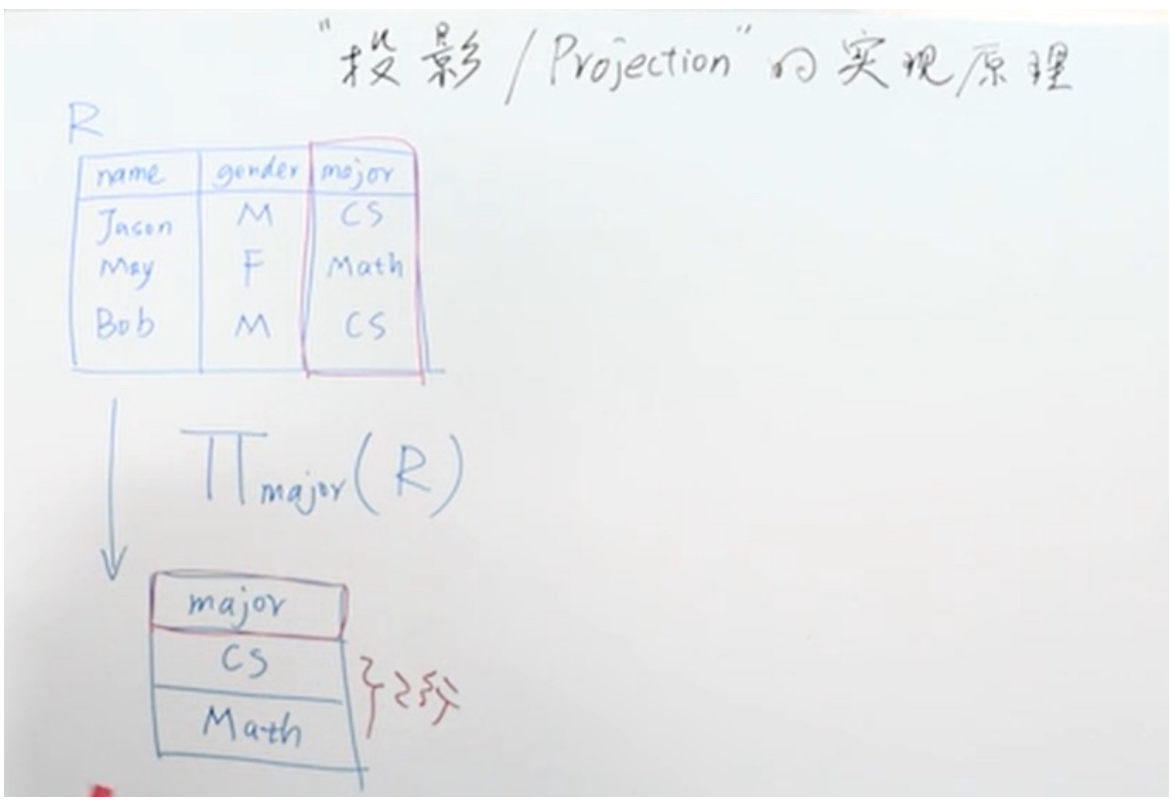
6.5 投影算子的实现

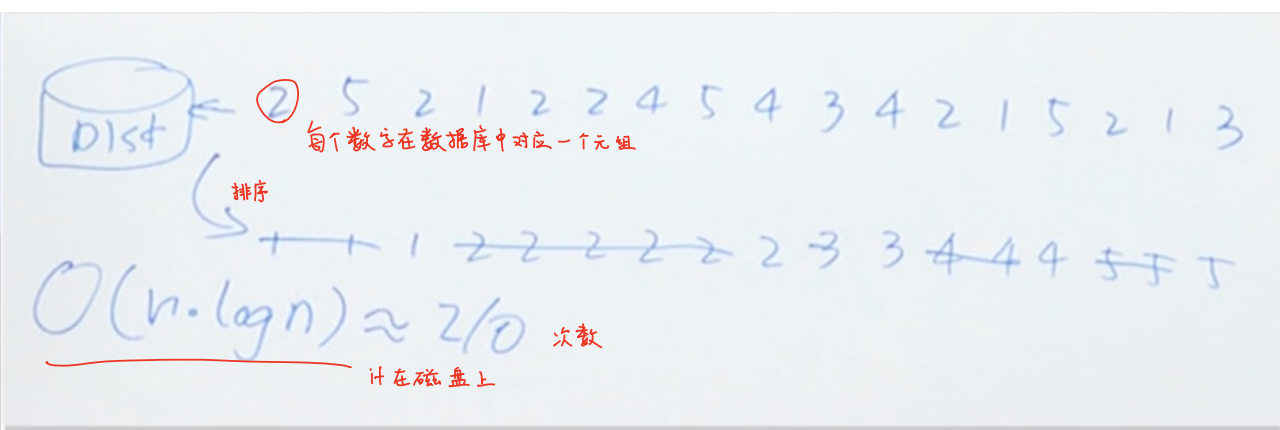
去重 需要满足关系的定义

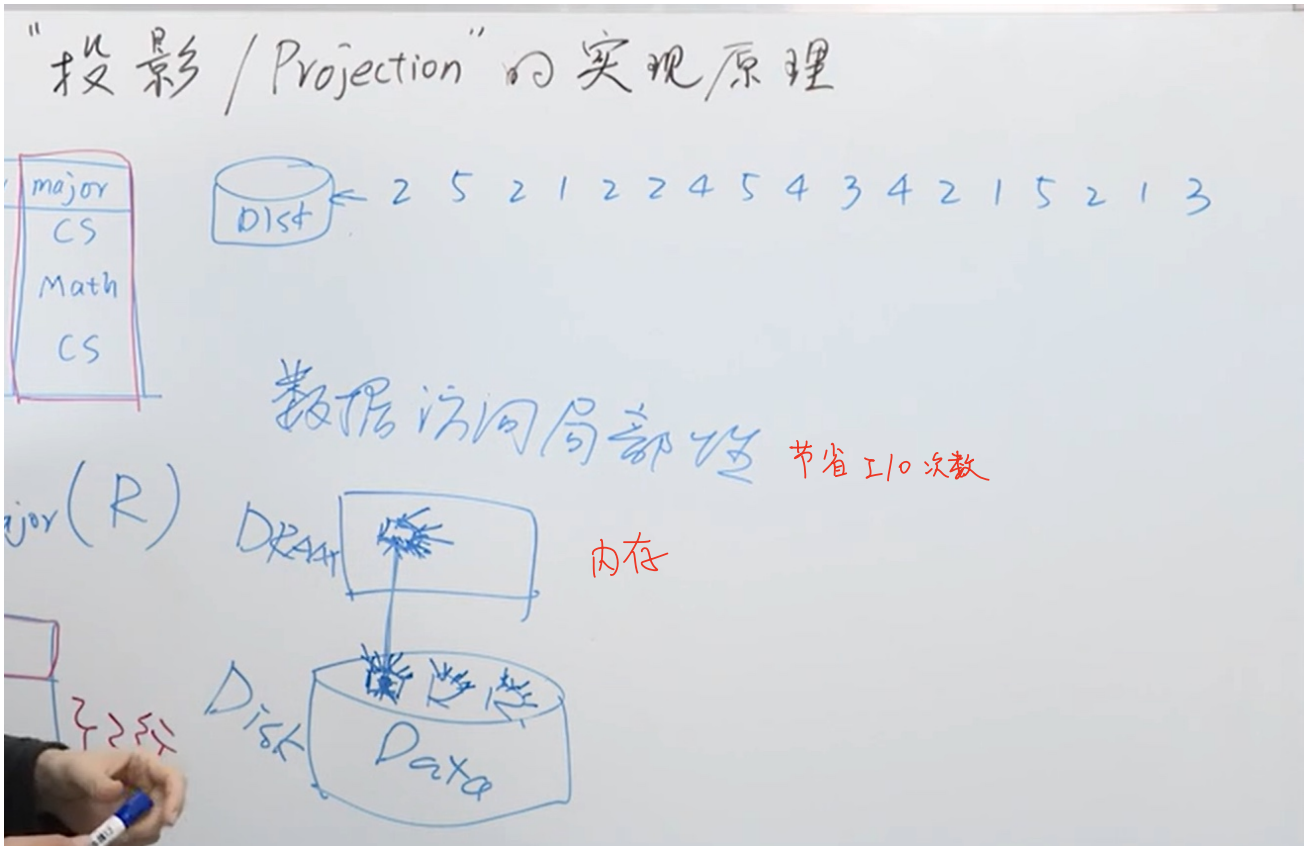
先做排序，假设数据在磁盘上，排序算法的复杂度是o(nlogn)，约等于I/o次数，效率很低。

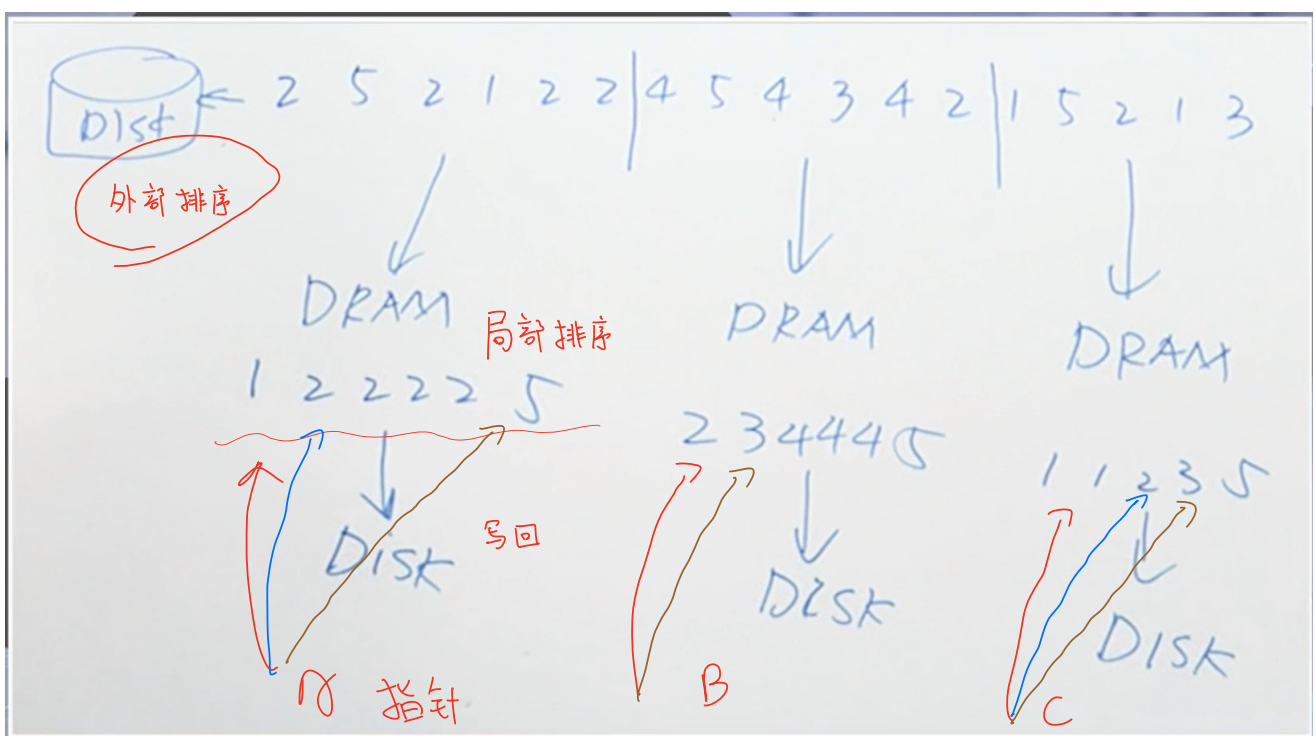
外部排序 考虑数据访问的局部性，数据在磁盘里面，内存作为

缓冲区放不下所有数据，访问数据的时候需要把页面从硬盘调入内存中，访问完再扔掉。局部性，如果我要访问若干次一个页面里的数据，我直接把这个页面导入内存，进行若干次访问，把该访问的都访问完，再从内存中丢弃这个页面。也就是说每次调入一个页面，你都尽可能的把页面所有的数据都处理之后再抛弃这个页面





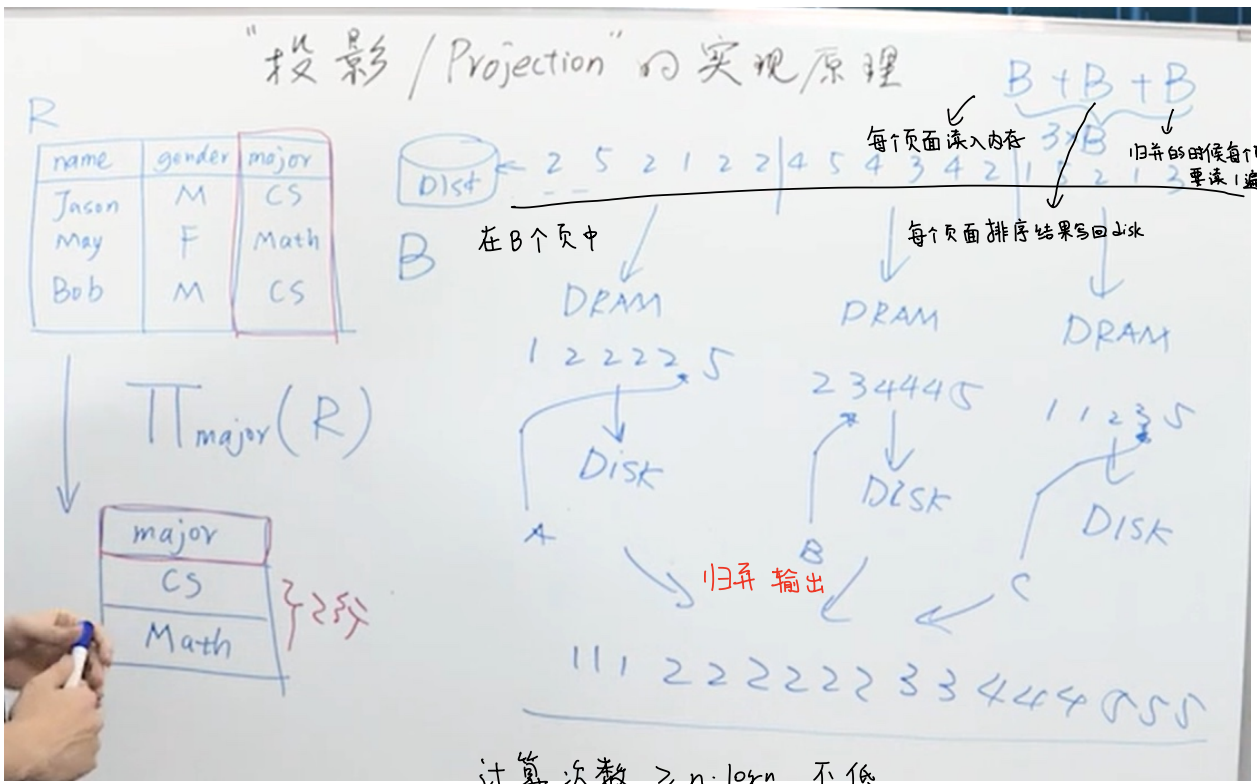




使用指针对硬盘上的数据作归并，再读入内存

1 1 1 2 2 2 2 2 2 3 3 4 4 4 5 5 5

排完就输出，占内存空间少



第 1 题(本题2分)：对于查询 Select Distinct sno, sname From Student, SC Where Student.sno=SC.sno And cno='C001'，请用常识判断以下哪个查询计划的执行效率应该最高？（假设SC在cno上有索引，Student在sno上有索引。）

Sno sname birth gender　Sno cno credit grade　找C001

太大了　取　连　取

- A: （1）Student和SC在sno上做连接；（2）在（1）的结果上做cno='C001'的选择；（3）将（2）的结果在sno, sname上做投影。 ✗
- B: （1）在SC上做cno='C001'的选择；（2）将（1）的结果和Student在sno上做连接；（3）将（2）的结果在sno, sname上做投影。 ✗
- C: （1）在SC上做cno='C001'的选择；（2）将student在sno, sname上做投影，将（1）的结果在sno上做投影；（3）将（2）的两项结果在sno上做连接。 ✗

不用全选，留下SC.sno即可　最后一步才用Index!?

- D: （1）在SC上做cno='C001'的选择；（2）将（1）的结果在sno上做投影；（3）将（2）的结果和Student在sno上做连接；（4）将（3）的结果在sno, sname上做投影。 ✓

连接的表小3但没必要，∵ Sno有Index，直接跟大表相连

Cno → SC_Sno → Student_Sno → Sno 和 Sname

第 2 题(本题2分)：对于一个关系代数表达的查询计划 (σ(A) ⋈ σ(B)) ⋈ C，它有几种等价的执行方式？（σ表示选择操作，⋈表示连接操作，这里A表和B表之间的连接属性与A表和B表之间的连接属性是不同的。）

选A　选B　C

C_5^2　5　5×4/2=10

- A: 2种 ✗
- B: 4种 ✗
- C: 10种 ✗
- D: 15种 ✓

A'　B'

AB　AC　BC

BA　CA　CB

3×

第 3 题(本题2分)：给定一张表R(a,b,c,d)，假设一个查询为Select a,b,c From R Where a=10 OR b=100。以下说法正确的是?

- A：索引对该查询无法提供帮助 ✗
- B：在a或b上创建索引都可加速该查询的执行 ✗
- C：在a和b上各创建一个索引可加速该查询的执行 ✓
- D：在a,b上创建一个复合索引可加速该查询的执行 ✗

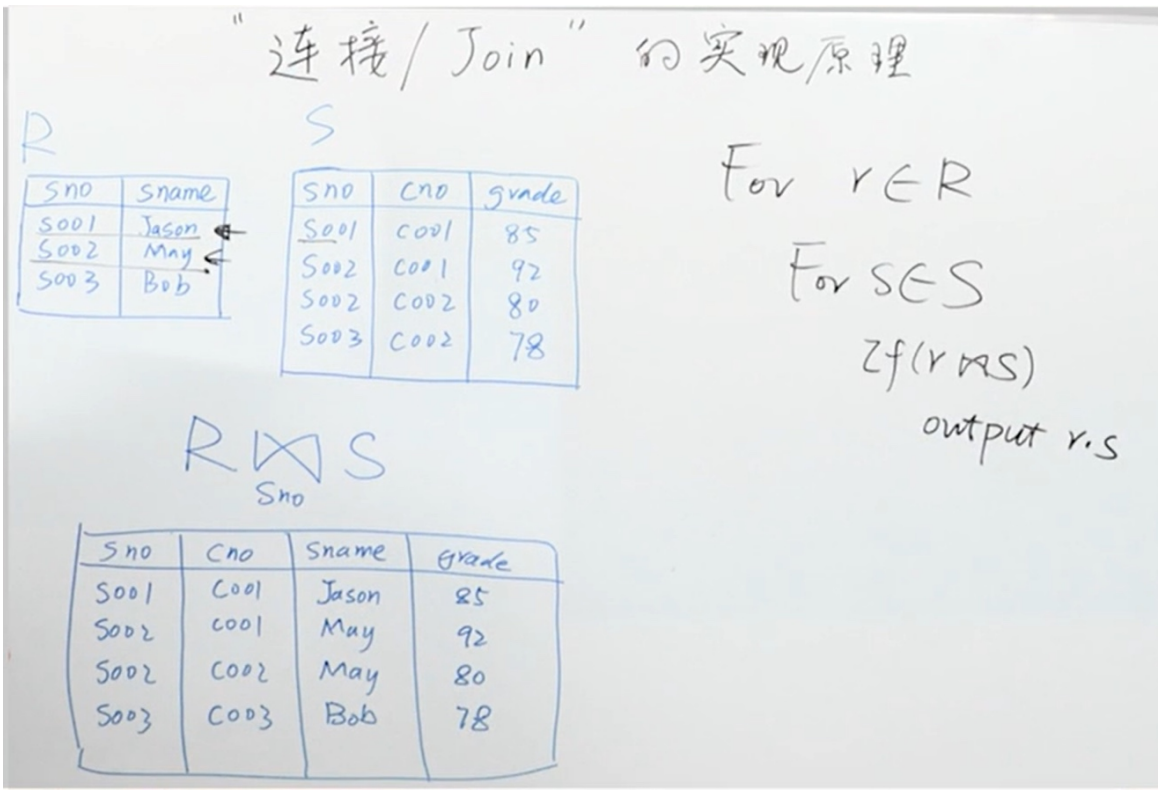
第 4 题(本题2分)：对于两个等价的查询计划 σ(A) ⋈ B 和 σ(A ⋈ B)，要判断它们哪个执行效率更高，以下哪些因素的作用最小。

- A：B上是否有索引 ✗
- B：A ⋈ B的结果集大小 ✗
- C：系统内存空间的大小 ✗
- D：系统硬盘空间的大小 ✓

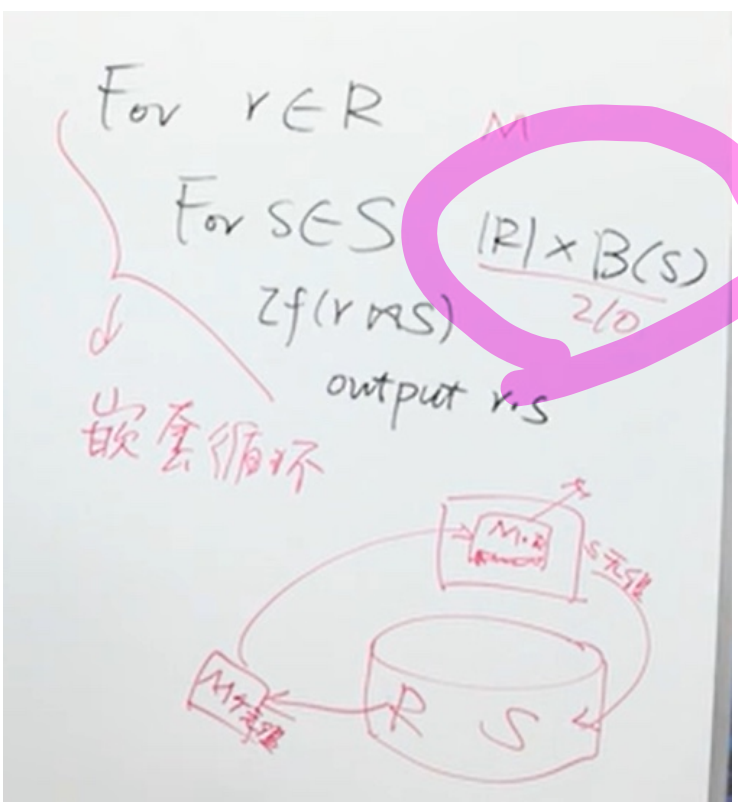
6.6 连接算子的实现 -嵌套循环

可以看成是两重的循环

第一个循环是对R表里的每一个元组进行考察，第二重循环是对R表里的每一个小r，尝试跟S表里的每一个元组s进行连接，最后如果r和s满足连接的要求，可以连接的话，就输出到结果表里。

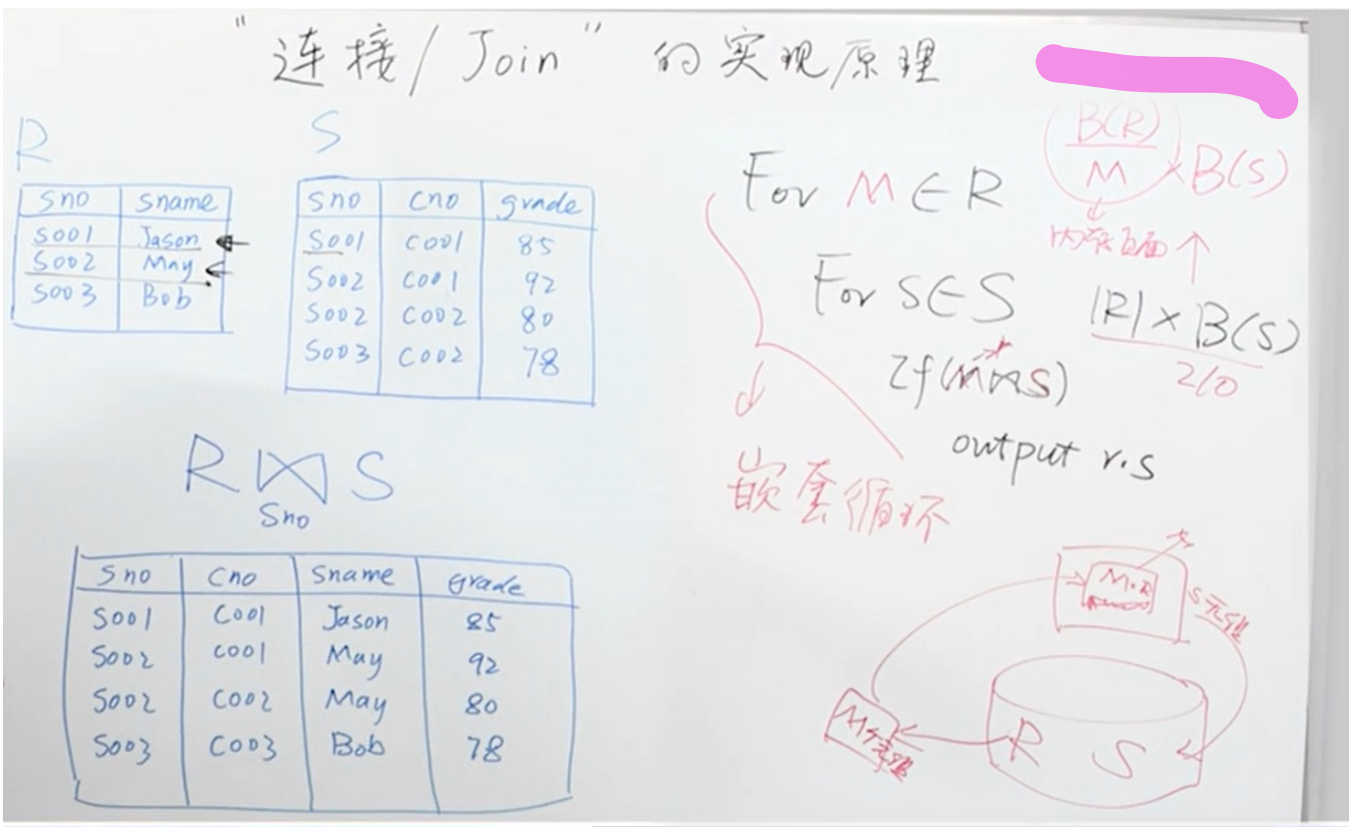


大概的I/O代价：（R）里面元组的个数乘以S表的页数，代价很高。R表里有多少元组就相当于要把S表扫描多少遍。效率很低。



嵌套循环：

不是对R里面的每一个元组都进行对S表的扫描，而是对R表里的M个元组进行S表的扫描。内存每次能装下R表的M个元组，再去扫描S表，对S表里的每一个元组进行与内存中M个元组的整体尝试（对M中的每一个小r）。I/O次数大大降低，B（R）代表装下R表的页面个数，M代表内存里面可以容纳的页面的个数。每放一大块到内存里面，我就对S表进行一个扫描，对内存里面所有的元组进行匹配



6.7 连接算子的实现 -散列连接

也叫哈希join hash join

对R表和S表进行预处理，使用哈希函数H（x），把连接属性的取值带入哈希函数里面，得到一个哈希值。这个哈希函数的目的是把连接属性的取值映射到一个个桶里面。每个桶对应的连接属性的取值（哈希函数值）是不一样的。可以得出结论：R表和S表的第x号桶，所对应的连接属性的取值范围是一样的。所以我们其实是把R表和S表进行了切分，R1桶里的元组只能和S1桶里的元组形成连接，这是由哈希计算的过程所保证的。得到预处理的结果以后，把Rx和Sx桶一次性调入内存里做连接（这实际上是对两张表又做了一次扫描）

I/O代价：

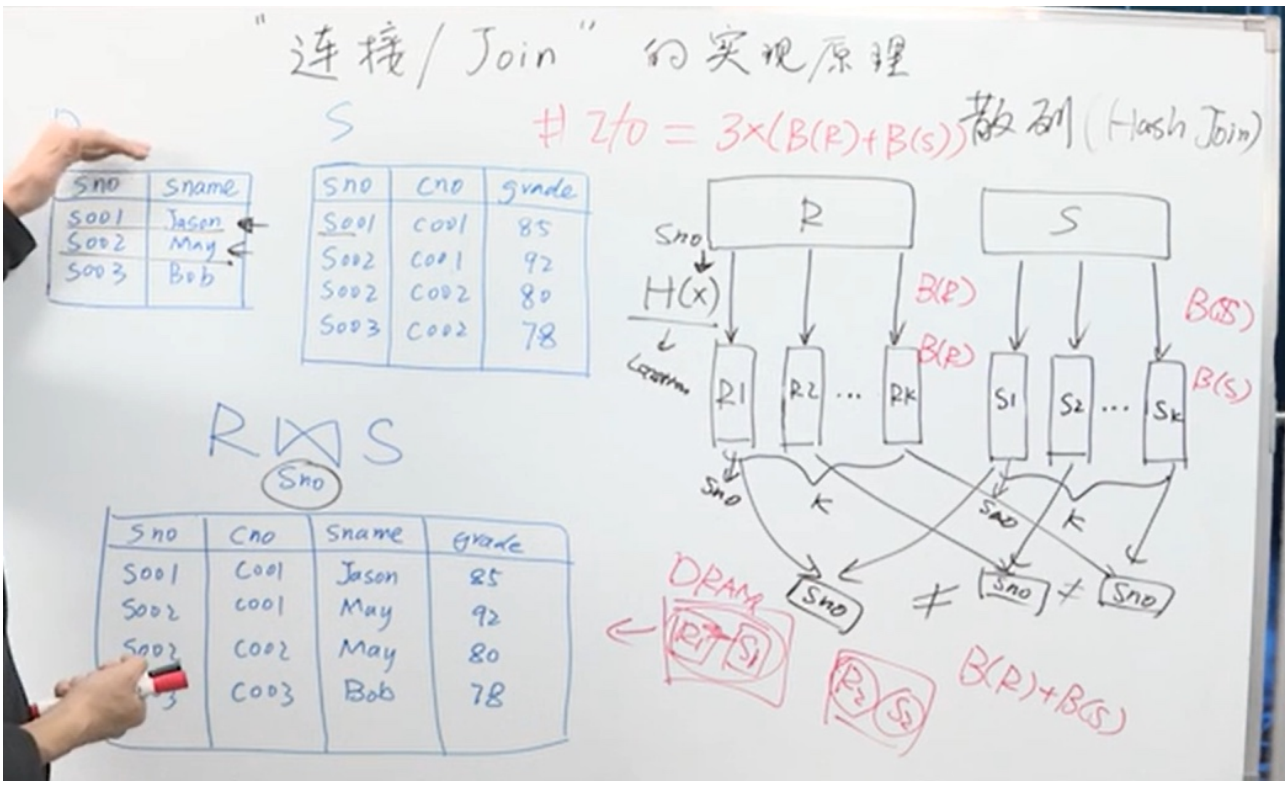
做散列操作，需要把R表和S表再读进去一遍，B（R）代表R表包含的页面的个数，B（S）代表S表包含的页面的个数。B（R）+B（S）次I/O消耗

把散列结果写出到硬盘桶里面，次数同样是B（R）+B（S）次

最后桶之间两两做连接，还得全部扫描一次，也是B（R）+B（S）次操作

所以总的来说，整个连接操作的I/O代价是3x（B（R）+B（S）次）。是一个相对小的代价，相当于你要做连接，把两张表扫描三遍代价

散列连接和嵌套循环哪个更好要看情况。



6.8 连接算子的实现- 索引连接 index join

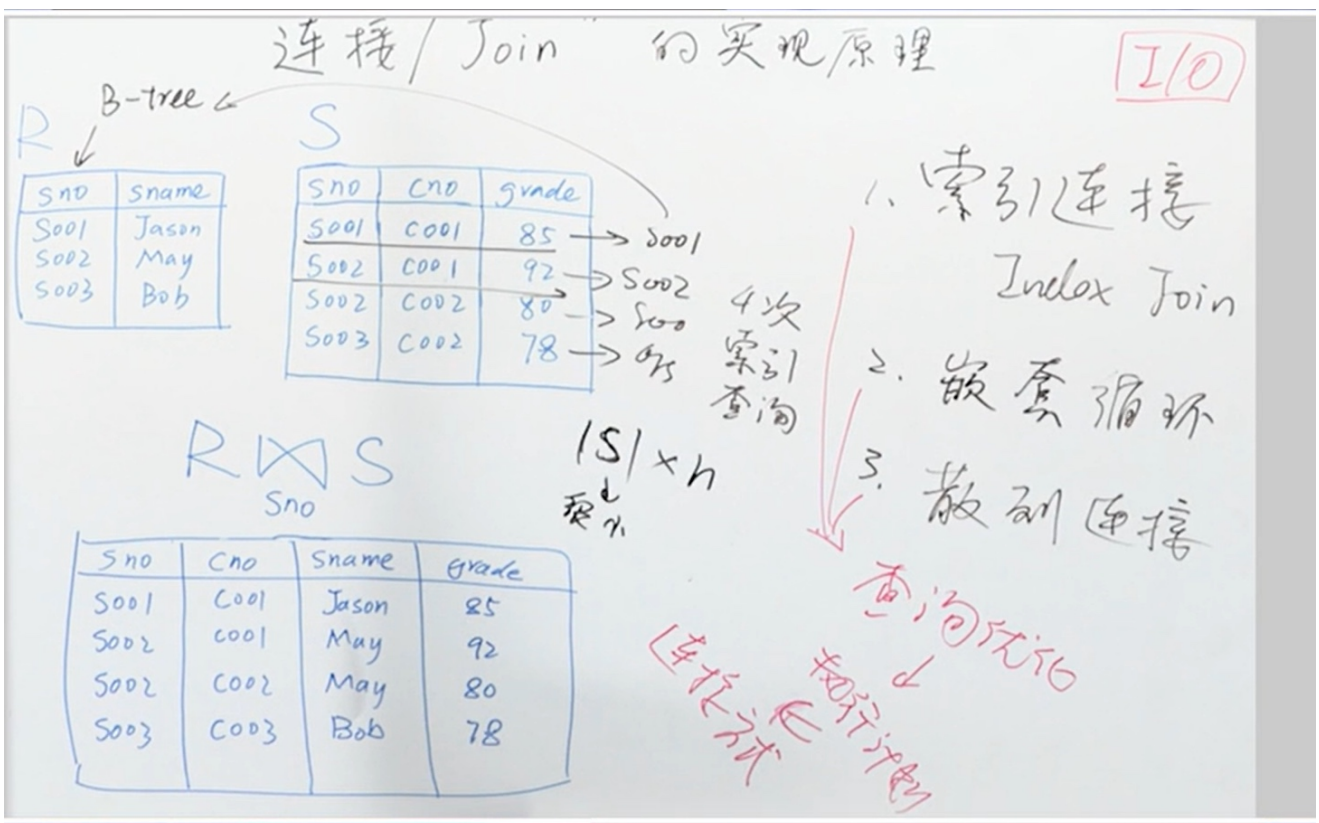
利用索引完成连接操作

假设在连接属性上做了一个B树索引。每拿S表的一个元组出来，利用R表的索引查自己表里面符合连接条件的 元组，然后完成对于这个元组的连接操作，直到扫描完S表。计算代价：对S表里的每一个元组都要做索引查询，假设索引查询每做一次，I/O代价是h，则总代价是S表中元组的个数乘以h，当S表很小的时候，这个代价是小的。对于一个数据库系统而言，当你在

某些连接属性上已经创建了索引，它会考虑是否使用索引连接是一个高效的选择。

使用哪种连接方式最好 是查询优化的一部分，由查询优化器判断

数据密集型算法的设计要重视数据访问I/O的代价



第 1 题(本题2分)：如课程中提到的，嵌套循环算法的I/O代价可以表示为 B(R) + B(S)*B(R)/M，其中B(R)代表外循环表的页数，B(S)代表内循环表的页数，M代表内存能容纳的页数。那么以下哪种方案无法提升该算法的效率?

- A：增加内存容量 ✗
- B：将小表放到内循环（作为S），大表放到外循环（作为R） ✓ B(R) 小点好
- C：将大表放到内循环（作为S），小表放到外循环（作为R） ✗
- D：更换速度更快的硬盘 ✗

第 2 题(本题2分)：课程中介绍了两种连接操作执行算法，嵌套循环和散列连接。以下哪种场景更适合使用嵌套循环算法?

- (●) A：一张表很大，一张表很小。小的表几乎可以容纳到内存中。 ✓
- ( ) B：两张表都比较大，都无法容纳到内存中。 ✗
- ( ) C：两张表都比较小，都可以容纳到内存中。 ✗
- ( ) D：几乎在所有情况下，散列连接的效率都更高。 ✗

第 3 题(本题2分)：假设R表有1000行，S表有100行；每页可以容纳10行数据（无论R表或S表）；内存可以容纳3页；R表和S表均有x属性，并且在x属性上均创建了索引；假设x具备很高的辨识度并且分布均匀。那么，理论上实现R和S在x上的等值连接的最佳算法是:

- ( ) A：嵌套循环 ✗
- ( ) B：散列连接 ✗
- (●) C：索引连接 ✓
- ( ) D：都一样 ✗

100页 10页 index join