**精益公司内部培训**

**索引、视图**

**一、从一个故事开始。。。**

**到底什么是索引？？**

对索引的直观理解：很久以前，在一个古城的大图书馆中珍藏有成千上万本书籍，但书架上的书没有按任何顺序摆放，因此每当有人询问某本书时，图书管理员只有挨个寻找，每一次都要花费大量的时间。

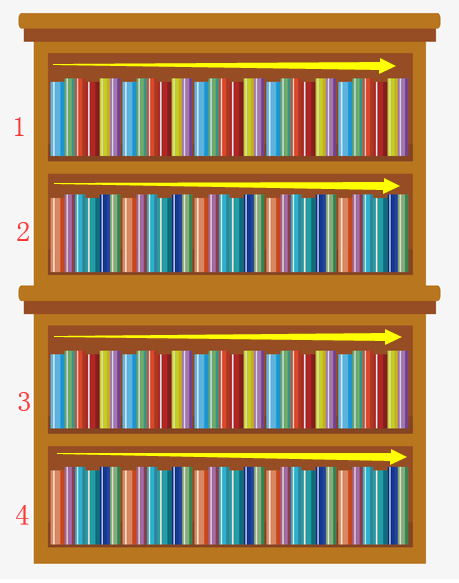
[这就好比数据表没有主键一样，搜索表中的数据时，数据库引擎必须进行全表扫描，效率极其低下。] ，更糟的是图书馆的图书越来越多，图书管理员的工作变得异常痛苦。

**最重要的索引：主键（一切索引的基础）**

有一天来了一个聪明的小伙子，他看到图书管理员的痛苦工作后，想出了一个办法，他建议将每本书都编上号，然后按编号放到书架上，如果有人指定了图书编号，那么图书管理员很快就可以找到它的位置了。

　　[给图书编号就像给表创建主键一样，**创建主键时，会创建聚集索引树**，表中的**所有行**会在文件系统上根据主键值进行物理排序，当查询表中任一行时，数据库首先使用聚集索引树找到对应的数据页(就像首先找到书架一样)，然后在数据页中根据主键键值找到目标行(就像找到书架上的书一样)。]

[在一个表上只能创建一个聚集索引，就像书只能按一种规则摆放一样。]



　　但问题并未完全解决，因为很多人记不住书的编号，只记得书的名字，图书管理员无奈又只有扫描所有的图书编号挨个寻找，但这次他只花了20分钟，以前未给图书编号时要花2-3小时，但与根据图书编号查找图书相比，时间还是太长了，因此他向那个聪明的小伙子求助。

[这就好像你给Product表增加了主键ProductID，但除此之外没有建立其它索引，当使用Product Name进行检索时，数据库引擎又只要进行全表扫描，逐个寻找了。]

创建非聚集索引：

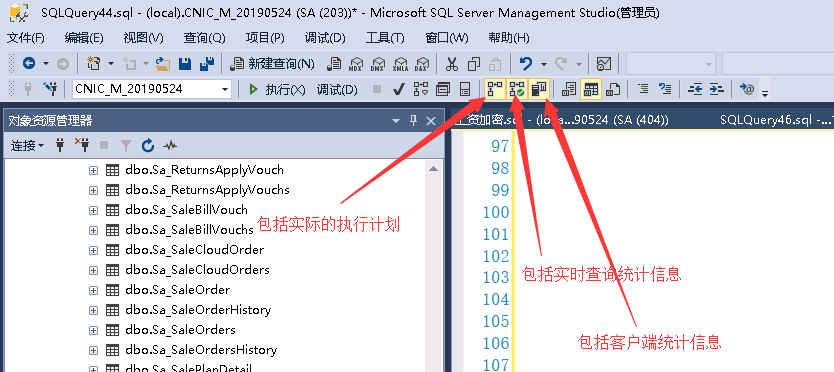
　　聪明的小伙告诉图书管理员，之前已经创建好了图书编号，现在只需要再创建一个索引或目录，将图书名称和对应的编号一起[存储](http://storage.it168.com/" \o "存储" \t "https://www.cnblogs.com/Shaina/archive/2012/04/22/_blank)起来，但这一次是按图书名称进行排序，如果有人想找“Database Management System”一书，你只需要跳到“D”开头的目录，然后按照编号就可以找到图书了。

　　于是图书管理员兴奋地花了几个小时创建了一个“图书名称”目录，经过测试，现在找一本书的时间缩短到1分钟了(其中30秒用于从“图书名称”目录中查找编号，另外根据编号查找图书用了30秒)。

　　图书管理员开始了新的思考，读者可能还会根据图书的其它属性来找书，如作者，于是他用同样的办法为作者也创建了目录，现在可以根据**图书编号**，书名和作者在1分钟内查找任何图书了，图书管理员的工作变得轻松了，故事也到此结束。

到此，我相信你已经完全理解了**索引的真正含义**。假设我们有一个Products表，创建了一个聚集索引(根据表的主键自动创建的)，我们还需要在ProductName列上创建一个非聚集索引，创建非聚集索引时，数据库引擎会为非聚集索引自动创建一个索引树(就像故事中的“图书名称”目录一样)，产品名称会存储在索引页中，每个索引页包括一定范围的产品名称和它们对应的主键键值，当使用产品名称进行检索时，数据库引擎首先会根据产品名称查找非聚集索引树查出主键键值，然后使用主键键值查找聚集索引树找到最终的产品。

1. 从故事到实践



--添加主键

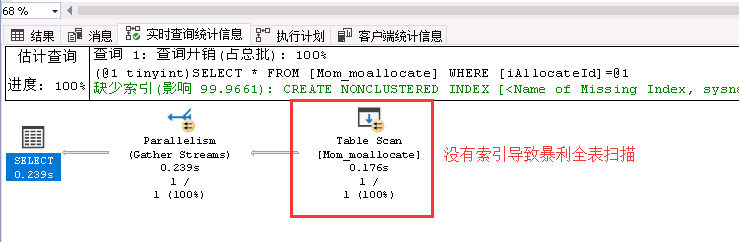
alter table 表名 add constraint 主键名 primary key(字段名1,字段名2……)

--删除主键

alter table表名 drop constraint 主键名

在没有主键的情况下进行查找

SELECT \* FROM Mom\_moallocate WHERE iAllocateId=79

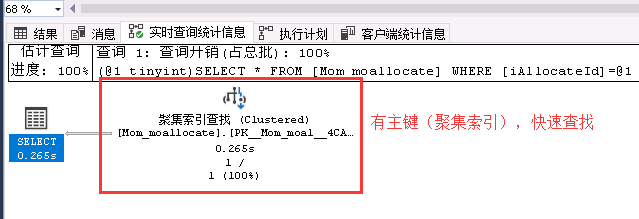


创建 Mom\_moallocate 表的主键iAllocateId

alter table Mom\_moallocate add constraint PK\_\_Mom\_moal\_\_4CABEA407AE30E2B primary key(iAllocateId)

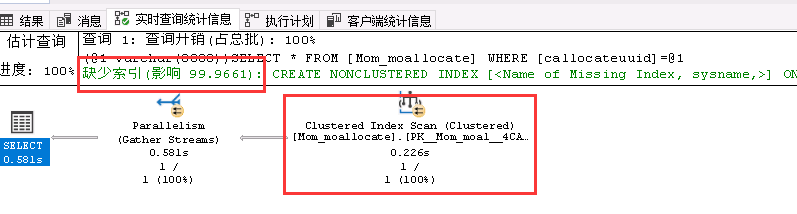
在有主键的情况下进行查找

SELECT \* FROM Mom\_moallocate WHERE iAllocateId=79

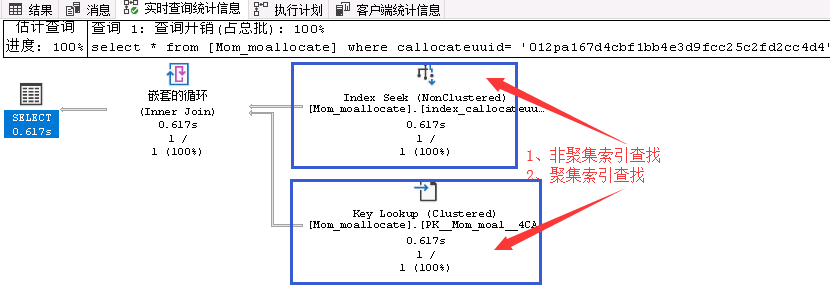


在没有非聚集索引的情况下进行查找

select \* from [Mom\_moallocate] where callocateuuid= '012pa167d4cbf1bb4e3d9fcc25c2fd2cc4d4'



　创建该字段的非聚集单列索引，然后进行查询

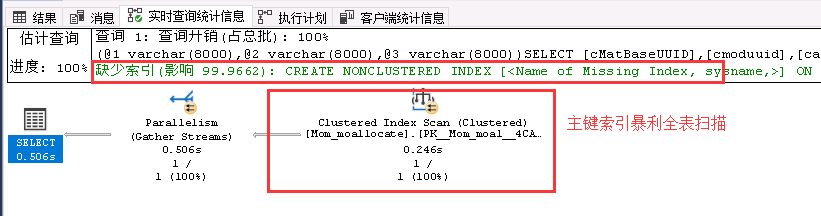


查找过程，首先根据 callocateuuid查找主键iAllocateId，然后根据主键iAllocateId查找相关内容。

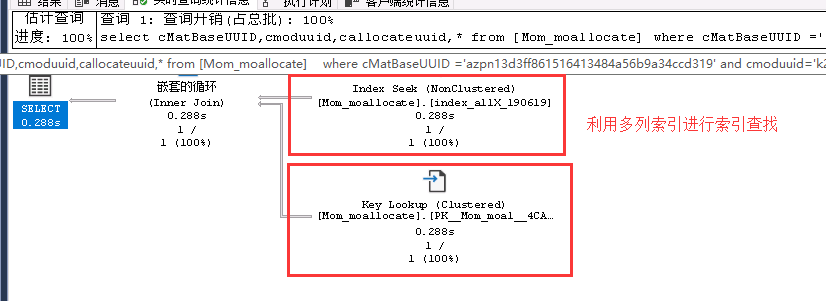
多列索引

在没有非聚集索引的情况下进行多列查询

select cMatBaseUUID,cmoduuid,callocateuuid,\* from [Mom\_moallocate] where cMatBaseUUID ='azpn13d3ff861516413484a56b9a34ccd319' and cmoduuid='k2o3d0817c4181bd4fa28909880639a349e1' and callocateuuid ='012pa167d4cbf1bb4e3d9fcc25c2fd2cc4d4'



create nonclustered index index\_allX\_190619 ON Mom\_moallocate(callocateuuid,cMatBaseUUID,allocateId)

再次查找

对多列索引的单列，两列组合进行查找（第一列除外），观察复合索引是否起作用

select cMatBaseUUID,cmoduuid,callocateuuid,\* from [Mom\_moallocate]

where callocateuuid= '012pa167d4cbf1bb4e3d9fcc25c2fd2cc4d4'

select cMatBaseUUID,cmoduuid,callocateuuid,\* from [Mom\_moallocate]

where cmoduuid='k2o3d0817c4181bd4fa28909880639a349e1'

select cMatBaseUUID,cmoduuid,callocateuuid,\* from [Mom\_moallocate]

where callocateuuid= '012pa167d4cbf1bb4e3d9fcc25c2fd2cc4d4'

and cmoduuid='k2o3d0817c4181bd4fa28909880639a349e1'

select cMatBaseUUID,cmoduuid,callocateuuid,\* from [Mom\_moallocate]

where cmoduuid='k2o3d0817c4181bd4fa28909880639a349e1'and

callocateuuid= '012pa167d4cbf1bb4e3d9fcc25c2fd2cc4d4'

三、 单列索引与多列索引的思考

考虑两种不同的建立索引方式：

　　case 1:对c1,c2,c3三列按此顺序添加一个多列索引;

　　case 2: 对c1,c2,c3分别建立三个单列索引;

　　问题1:按c1搜索时，哪种索引效率快?

　　答：case2

　　问题2:按C2搜索时，哪种索引效率快?

　　答：case2,并且，case1的索引无效

　　问题3:按C1,C2搜索时哪种效率快?

　　答：不知道

　　问题四：按C1,C2,C3搜索哪种效率快?

　　答：case1

　　问题五：按C2,C3,C1搜索时哪种效率快?

答：case2,因为没有按多列索引的顺序搜索，case1的索引没有使用到。

问题六：如果没有case 2，分别搜索C1,C2,C3，复合索引是否起作用

　　四、 单个表中索引太多的负面影响

　　当一个表存在多个(单列)索引，将造成Delete ,update ,insert操作需要花费大量的时间删除索引和重建索引。

　　通过把多个(单列)索引合并成一个(多列)索引后，测试得出Delete ,update ,insert操作时需要花费的时间大大缩短。但是这样可能会对之前单列索引字段的查询性能有影响。个中好处，权衡取舍。

五、如何查看已有索引？ sp\_helpindex 表名

视图

1.解释

视图从代码上看是一个select语句

视图从逻辑上看是一个虚拟表,视图里无法执行insert，update,delete操作

2.格式

create view 视图的名字

as

select语句

作业，在**模拟库**找一个大视图，创建相应的索引。（提示，先用sp\_helpindex查看原来的索引）