SQLServer索引结构及其使用（一） PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/179/2021_2022_SQLServer_E7_c98_179405.htm 计算机等级考试训练软件《百宝箱》一、深入浅出理解索引结构 实际上，您可以把索引理解为一种特殊的目录。微软的SQL SERVER提供了两种索引：聚集索引（clustered index，也称聚类索引、簇集索引）和非聚集索引（nonclustered index，也称非聚类索引、非簇集索引）。下面，我们举例来说明一下聚集索引和非聚集索引的区别： 其实，我们的汉语字典的正文本身就是一个聚集索引。比如，我们要查“ 安 ”字，就会很自然地翻开字典的前几页，因为“ 安 ”的拼音是“ an ”，而按照拼音排序汉字的字典是以英文字母“ a ”开头并以“ z ”结尾的，那么“ 安 ”字就自然地排在字典的前部。如果您翻完了所有以“ a ”开头的部分仍然找不到这个字，那么就说明您的字典中没有这个字；同样的，如果查“ 张 ”字，那您也会将您的字典翻到最后部分，因为“ 张 ”的拼音是“ zhang ”。也就是说，字典的正文部分本身就是一个目录，您不需要再去查其他目录来找到您需要找的内容。我们把这种正文内容本身就是一种按照一定规则排列的目录称为“ 聚集索引 ”。 如果您认识某个字，您可以快速地从自动中查到这个字。但您也可能会遇到您不认识的字，不知道它的发音，这时候，您就不能按照刚才的方法找到您要查的字，而需要去根据“ 偏旁部首 ”查到您要找的字，然后根据这个字后的页码直接翻到某页来找到您要找的字。但您结合“ 部首目录 ”和“ 检字表 ”而查到的字的排序并不是真正的正文的排序方法，比如您查“ 张 ”字，我们可以看到

在查部首之后的检字表中“ 张 ”的页码是672页，检字表中“ 张 ”的上面是“ 驰 ”字，但页码却是63页，“ 张 ”的下面是“ 弩 ”字，页面是390页。很显然，这些字并不是真正的分别位于“ 张 ”字的上下方，现在您看到的连续的“ 驰、张、弩 ”三字实际上就是他们在非聚集索引中的排序，是字典正文中的字在非聚集索引中的映射。我们可以通过这种方式来找到您所需要的字，但它需要两个过程，先找到目录中的结果，然后再翻到您所需要的页码。我们把这种目录纯粹是目录，正文纯粹是正文的排序方式称为“ 非聚集索引 ”。 通过以上例子，我们可以理解到什么是“ 聚集索引 ”和“ 非聚集索引 ”。进一步引申一下，我们可以很容易的理解：每个表只能有一个聚集索引，因为目录只能按照一种方法进行排序。二、何时使用聚集索引或非聚集索引下面的表总结了何时使用聚集索引或非聚集索引（很重要）： 动作描述 使用聚集索引 使用非聚集索引 列经常被分组排序 应 应 返回某范围内的数据 应 不应 一个或极少不同值 不应 不应 小数目的不同值 应 不应 大数目的不同值 不应 应 频繁更新的列 不应 应 外键列 应 应 主键列 应 应 频繁修改索引列 不应 应 事实上，我们可以通过前面聚集索引和非聚集索引的定义的例子来理解上表。如：返回某范围内的数据一项。比如您的某个表有一个时间列，恰好您把聚合索引建立在了该列，这时您查询2004年1月1日至2004年10月1日之间的全部数据时，这个速度就将是很快的，因为您的这本字典正文是按日期进行排序的，聚类索引只需要找到要检索的所有数据中的开头和结尾数据即可；而不像非聚集索引，必须先查到目录中查到每一项数据对应的页码，然后再根据页码查到具体内容。三、结合实际，

谈索引使用的误区 理论的目的是应用。虽然我们刚才列出了何时应使用聚集索引或非聚集索引，但在实践中以上规则却很容易被忽视或不能根据实际情况进行综合分析。下面我们将根据在实践中遇到的实际问题来谈一下索引使用的误区，以便于大家掌握索引建立的方法。1、主键就是聚集索引 这种想法笔者认为是极端错误的，是对聚集索引的一种浪费。虽然SQL SERVER默认是在主键上建立聚集索引的。 通常，我们会在每个表中都建立一个ID列，以区分每条数据，并且这个ID列是自动增大的，步长一般为1。我们的这个办公自动化的实例中的列Gid就是如此。此时，如果我们将这个列设为主键，SQL SERVER会将此列默认为聚集索引。这样做有好处，就是可以让您的数据在数据库中按照ID进行物理排序，但笔者认为这样做意义不大。 显而易见，聚集索引的优势是很明显的，而每个表中只能有一个聚集索引的规则，这使得聚集索引变得更加珍贵。 从我们前面谈到的聚集索引的定义我们可以看出，使用聚集索引的最大好处就是能够根据查询要求，迅速缩小查询范围，避免全表扫描。在实际应用中，因为ID号是自动生成的，我们并不知道每条记录的ID号，所以我们很难在实践中用ID号来进行查询。这就使让ID号这个主键作为聚集索引成为一种资源浪费。其次，让每个ID号都不同的字段作为聚集索引也不符合“ 大数目的不同值情况下不应建立聚合索引 ”规则；当然，这种情况只是针对用户经常修改记录内容，特别是索引项的时候会负作用，但对于查询速度并没有影响。 在办公自动化系统中，无论是系统首页显示的需要用户签收的文件、会议还是用户进行文件查询等任何情况下进行数据查询都离不开字段的是“ 日期 ”还有用户

本身的“ 用户名 ”。 通常，办公自动化的首页会显示每个用户尚未签收的文件或会议。虽然我们的where语句可以仅仅限制当前用户尚未签收的情况，但如果您的系统已建立了很长时间，并且数据量很大，那么，每次每个用户打开首页的时候都进行一次全表扫描，这样做意义是不大的，绝大多数的用户1个月前的文件都已经浏览过了，这样做只能徒增数据库的开销而已。事实上，我们完全可以让用户打开系统首页时，数据库仅仅查询这个用户近3个月来未阅览的文件，通过“ 日期 ”这个字段来限制表扫描，提高查询速度。如果您的办公自动化系统已经建立的2年，那么您的首页显示速度理论上将是原来速度8倍，甚至更快。 在这里之所以提到“ 理论上 ”三字，是因为如果您的聚集索引还是盲目地建在ID这个主键上时，您的查询速度是没有这么高的，即使您在“ 日期 ”这个字段上建立的索引（非聚合索引）。下面我们就来看一下在1000万条数据量的情况下各种查询的速度表现（3个月内的数据为25万条）：（1）仅在主键上建立聚集索引，并且不划分时间段：Select gid,fariqi,neibuyonghu,title from tgongwen用时：128470毫秒（即：128秒）（2）在主键上建立聚集索引，在fariq上建立非聚集索引：0select gid,fariqi,neibuyonghu,title from Tgongwenwhere fariqi> dateadd(day,-90,getdate())用时：53763毫秒（54秒）（3）将聚合索引建立在日期列（fariqi）上：0select gid,fariqi,neibuyonghu,title from Tgongwenwhere fariqi> dateadd(day,-90,getdate())用时：2423毫秒（2秒） 虽然每条语句提取出来的都是25万条数据，各种情况的差异却是巨大的，特别是将聚集索引建立在日期列时的差异。事实上，如果您的数据库真的有1000万容量的话，把主键建立在ID

列上，就像以上的第1、2种情况，在网页上的表现就是超时，根本就无法显示。这也是我摒弃ID列作为聚集索引的一个最重要的因素。得出以上速度的方法是：在各个0select语句前加：declare @d datetimeset @d=getdate()并在0select语句后加：0select [语句执行花费时间(毫秒)]=datediff(ms,@d,getdate())2、只要建立索引就能显著提高查询速度 事实上，我们可以发现上面的例子中，第2、3条语句完全相同，且建立索引的字段也相同；不同的仅是前者在fariqi字段上建立的是非聚合索引，后者在此字段上建立的是聚合索引，但查询速度却有着天壤之别。所以，并非是在任何字段上简单地建立索引就能提高查询速度。 从建表的语句中，我们可以看到这个有着1000万数据的表中fariqi字段有5003个不同记录。在此字段上建立聚合索引是再合适不过了。在现实中，我们每天都会发几个文件，这几个文件的发文日期就相同，这完全符合建立聚集索引要求的：“ 既不能绝大多数都相同，又不能只有极少数相同 ”的规则。由此看来，我们建立“ 适当 ”的聚合索引对于我们提高查询速度是非常重要的。3、把所有需要提高查询速度的字段都加进聚集索引，以提高查询速度 上面已经谈到：在进行数据查询时都离不开字段的是“ 日期 ”还有用户本身的“ 用户名 ”。既然这两个字段都是如此的重要，我们可以把他们合并起来，建立一个复合索引（compound index）。 很多人认为只要把任何字段加进聚集索引，就能提高查询速度，也有人感到迷惑：如果把复合的聚集索引字段分开查询，那么查询速度会减慢吗？带着这个问题，我们来看一下以下的查询速度（结果集都是25万条数据）：（日期列fariqi首先排在复合聚集索引的起始列，用户名neibuyonghu

排在后列）：100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com