SQLServer索引结构及其使用（三） 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_179403.htm 计算机等级考试训练软件《百宝箱》实现小数据量和海量数据的通用分页显示存储过程 建立一个 Web 应用，分页浏览功能必不可少。这个问题是数据库处理中十分常见的问题。经典的数据分页方法是:ADO 纪录集分页法，也就是利用ADO自带的分页功能（利用游标）来实现分页。但这种分页方法仅适用于较小数据量的情形，因为游标本身有缺点：游标是存放在内存中，很费内存。游标一建立，就将相关的记录锁住，直到取消游标。游标提供了对特定集合中逐行扫描的手段，一般使用游标来逐行遍历数据，根据取出数据条件的不同进行不同的操作。而对于多表和大表中定义的游标（大的数据集合）循环很容易使程序进入一个漫长的等待甚至死机。 更重要的是，对于非常大的数据模型而言，分页检索时，如果按照传统的每次都加载整个数据源的方法是非常浪费资源的。现在流行的分页方法一般是检索页面大小的块区的数据，而非检索所有的数据，然后单步执行当前行。 最早较好地实现这种根据页面大小和页码来提取数据的方法大概就是“ 俄罗斯存储过程 ”。这个存储过程用了游标，由于游标的局限性，所以这个方法并没有得到大家的普遍认可。 后来，网上有人改造了此存储过程，下面的存储过程就是结合我们的办公自动化实例写的分页存储过程：
CREATE procedure pagination1(@pagesize int, --页面大小，如每页存储20条记录@pageindex int --当前页码)asset nocount onbegindeclare @indextable table(id int identity(1,1),nid int) --定义

表变量declare @PageLowerBound int --定义此页的底码declare @PageUpperBound int --定义此页的顶码set @PageLowerBound=(@pageindex-1)*@pagesizeset @PageUpperBound=@PageLowerBound @pagesizeset rowcount @PageUpperBoundinsert into @indextable(nid) 0select gid from TGongwen where fariqi >dateadd(day,-365,getdate()) order by fariqi desc0select O.gid,O.mid,O.title,O.fadanwei,O.fariqi from TGongwen O,@indextable t where O.gid=t.nid and t.id>@PageLowerBound and t.id 以上存储过程运用了SQL SERVER的最新技术——表变量。应该说这个存储过程也是一个非常优秀的分页存储过程。当然，在这个过程中，您也可以把其中的表变量写成临时表：CREATE TABLE #Temp。但很明显，在SQL SERVER中，用临时表是没有用表变量快的。所以笔者刚开始使用这个存储过程时，感觉非常的不错，速度也比原来的ADO的好。但后来，我又发现了比此方法更好的方法。 笔者曾在网上看到了一篇小短文《从数据表中取出第n条到第m条的记录的方法》，全文如下：从publish 表中取出第 n 条到第 m 条的记录： SELECT TOP m-n 1 * FROM publish WHERE (id NOT IN (SELECT TOP n-1 id FROM publish)) id 为publish 表的关键字 我当时看到这篇文章的时候，真的是精神为之一振，觉得思路非常得好。等到后来，我在作办公自动化系统（ASP.NET C#＋SQL SERVER）的时候，忽然想起了这篇文章，我想如果把这个语句改造一下，这就可能是一个非常好的分页存储过程。于是我就满网上找这篇文章，没想到，文章还没找到，却找到了一篇根据此语句写的一个分页存储过程，这个存储过程也是目前较为流行的

一种分页存储过程，我很后悔没有争先把这段文字改造成存储过程：CREATE PROCEDURE pagination2(@SQL nVARCHAR(4000), --不带排序语句的SQL语句@Page int, --页码@RecsPerPage int, --每页容纳的记录数@ID VARCHAR(255), --需要排序的不重复的ID号@Sort VARCHAR(255) --排序字段及规则)ASDECLARE @Str nVARCHAR(4000)SET @Str=SELECT TOP CAST(@RecsPerPage AS VARCHAR(20)) * FROM (@SQL) T WHERE T. @ID NOT IN (SELECT TOP CAST((@RecsPerPage*(@Page-1)) AS VARCHAR(20)) @ID FROM (@SQL) T9 ORDER BY @Sort) ORDER BY @SortPRINT @StrEXEC sp_ExecuteSql @StrGO其实，以上语句可以简化为：SELECT TOP 页大小 *FROM Table1 WHERE (ID NOT IN (SELECT TOP 页大小*页数 id FROM 表 ORDER BY id))ORDER BY ID但这个存储过程有一个致命的缺点，就是它含有NOT IN字样。虽然我可以把它改造为：SELECT TOP 页大小 *FROM Table1 WHERE not exists(0select * from (0select top (页大小*页数) * from table1 order by id) b where b.id=a.id)order by id 即，用not exists来代替not in，但我们前面已经谈过了，二者的执行效率实际上是没有区别的。既便如此，用TOP 结合NOT IN的这个方法还是比用游标要来得快一些。 虽然用not exists并不能挽救上个存储过程的效率，但使用SQL SERVER中的TOP关键字却是一个非常明智的选择。因为分页优化的最终目的就是避免产生过大的记录集，而我们在前面也已经提到了TOP的优势，通过TOP 即可实现对数据量的控制。 在分页算法中，影响我们查询速度的关键因素有两点：TOP和NOT IN。TOP可以提高我们的查询速度，而NOT

IN会减慢我们的查询速度，所以要提高我们整个分页算法的速度，就要彻底改造NOT IN，同其他方法来替代它。 我们知道，几乎任何字段，我们都可以通过max(字段)或min(字段)来提取某个字段中的最大或最小值，所以如果这个字段不重复，那么就可以利用这些不重复的字段的max或min作为分水岭，使其成为分页算法中分开每页的参照物。在这里，我们可以用操作符“ > ”或“ Select top 10 * from table1 where id>200 于是就有了如下分页方案：0select top 页大小 *from table1 where id>(0select max (id) from (0select top ((页码-1)*页大小) id from table1 order by id) as T) order by id 在选择即不重复值，又容易分辨大小的列时，我们通常会选择主键。下表列出了笔者用有着1000万数据的办公自动化系统中的表，在以GID（GID是主键，但并不是聚集索引。）为排序列、提取gid,fariqi,title字段，分别以第1、10、100、500、1000、1万、10万、25万、50万页为例，测试以上三种分页方案的执行速度：（单位：毫秒）页码 方案1 方案2 方案3 1 60 30 76 10 46 16 63 100 1076 720 130 500 540 12943 83 1000 17110 470 250 10000 24796 4500 140 100000 38326 42283 1553 250000 28140 128720 2330 500000 121686 127846 7168 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