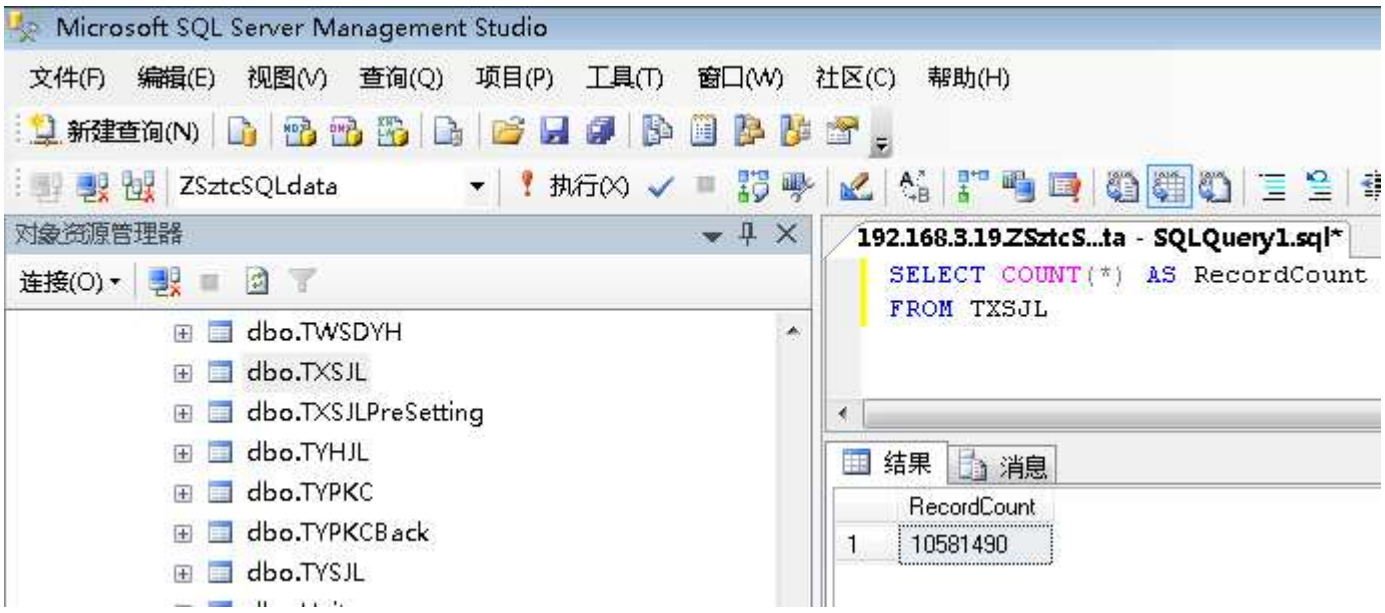
SqlSever2005 一千万条以上记录分页数据库优化经验总结【索引优化 + 代码优化】一周搞定

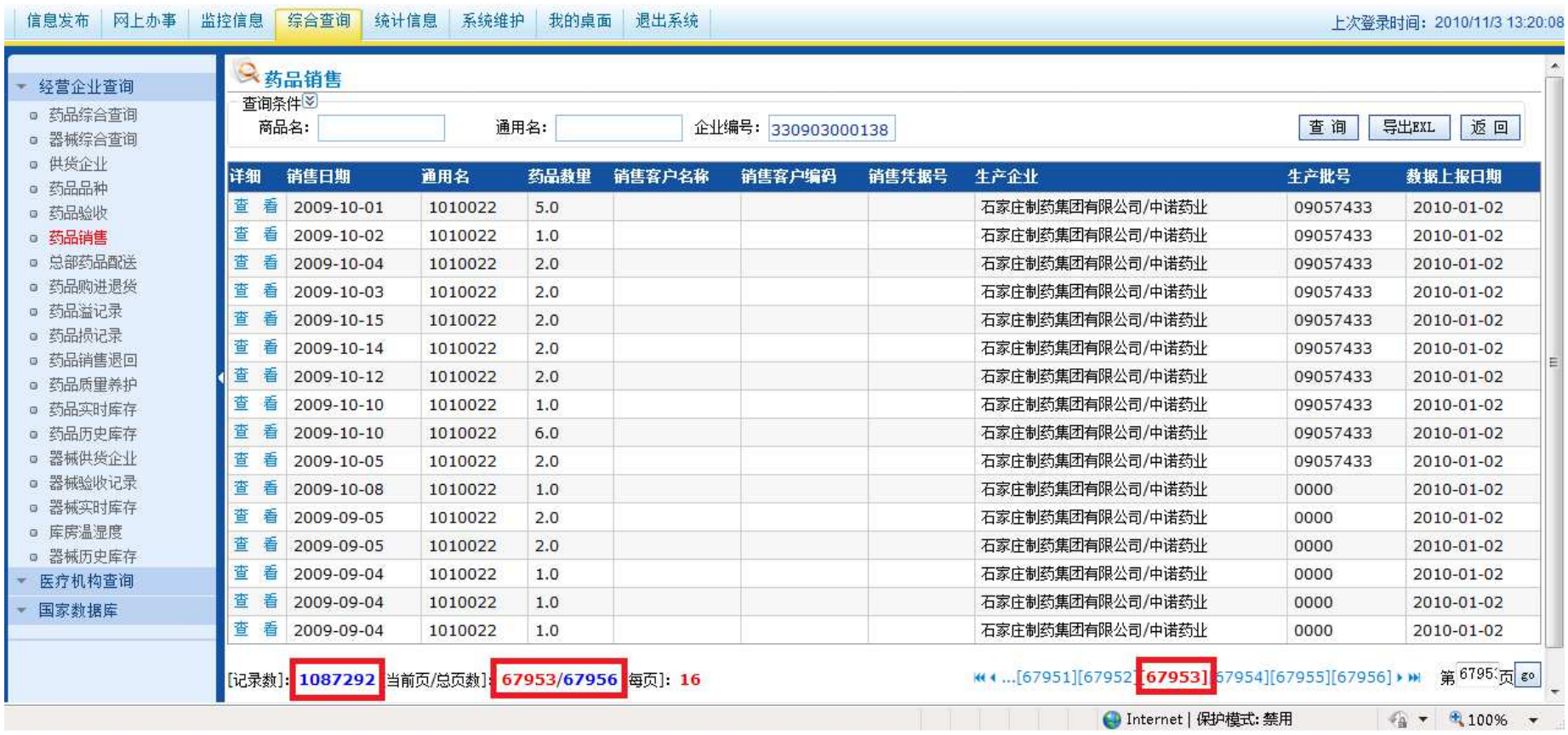
对普通开发人员来说经常能接触到上千万条数据优化的机会也不是很多，这里还是要感谢公司提供了这样的一个环境，而且公司让我来做优化工作。当数据库中的记录不超过 10 万条时，很难分辨出开发人员的水平有多高，当数据库中的记录条数超过 1000 万条后，还是蛮能考验开发人员的综合技术能力。

当然不是每个公司都能请得起专业的 DBA，话又说过来专业的 DBA 也未必能来我们公司长期工作，这就不只是薪资待遇问题了还会涉及到人家的长期发展规划了，当然我也不是专业的 DBA，本着能把问题解决好就是好猫的理念。

我们先看图，数据库中的记录数如下：记录数为 10581490 条同事还需要从另外一个表读取 7 万多条数据。



页面运行效果如下： 这是查看某个单位的数据，每页显示 16 条、记录数 1087292 条、分页数为 67956 页。



遇到的难题如下：

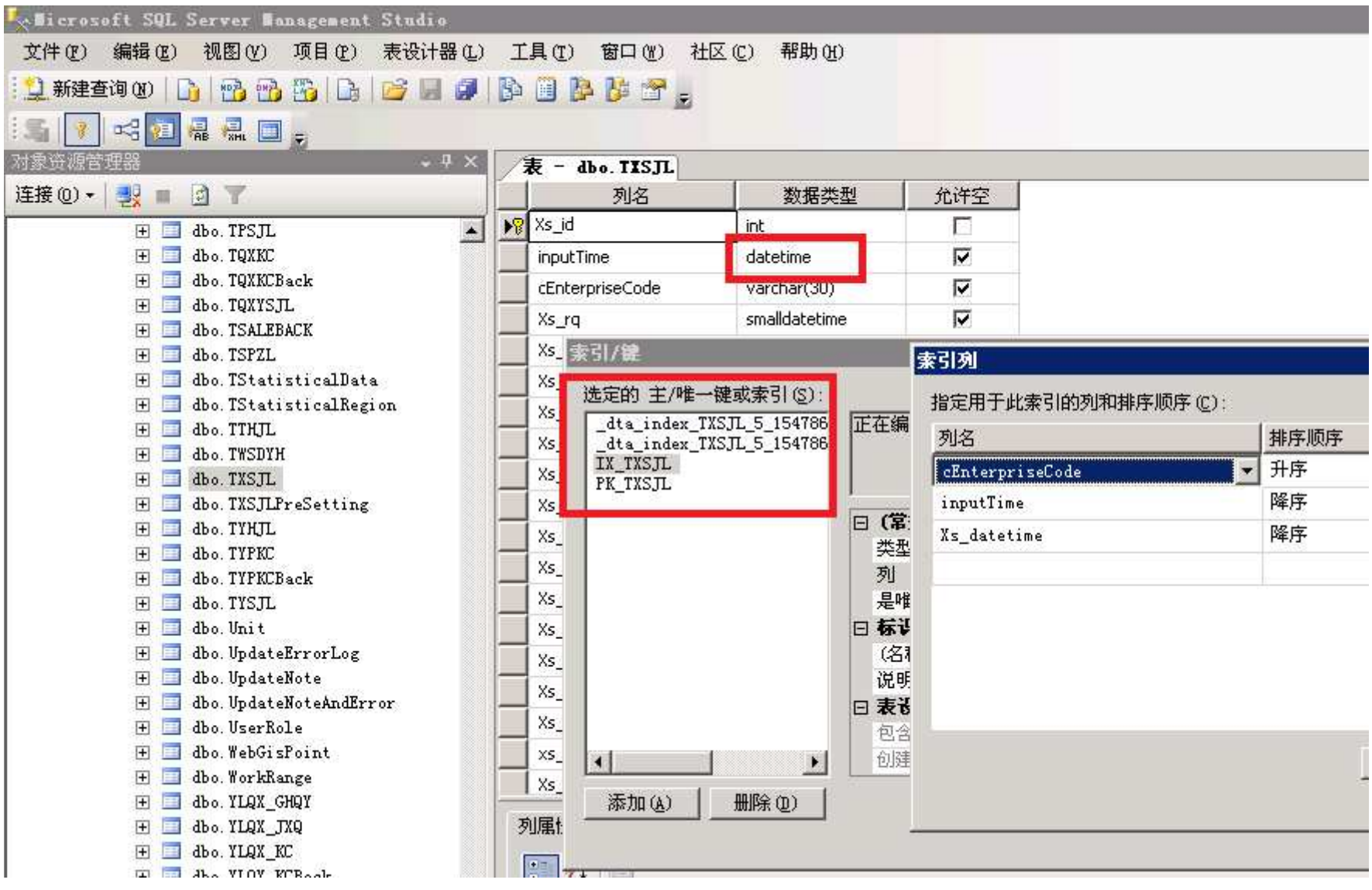
1：当客户用了几年后数据变得很庞大分页速度缓慢得要命几乎到了无法忍受的程度。

2：分页到最后一页时往往速度很慢会有死机现象出现，特别是记录条数很多时死机现象比较多。

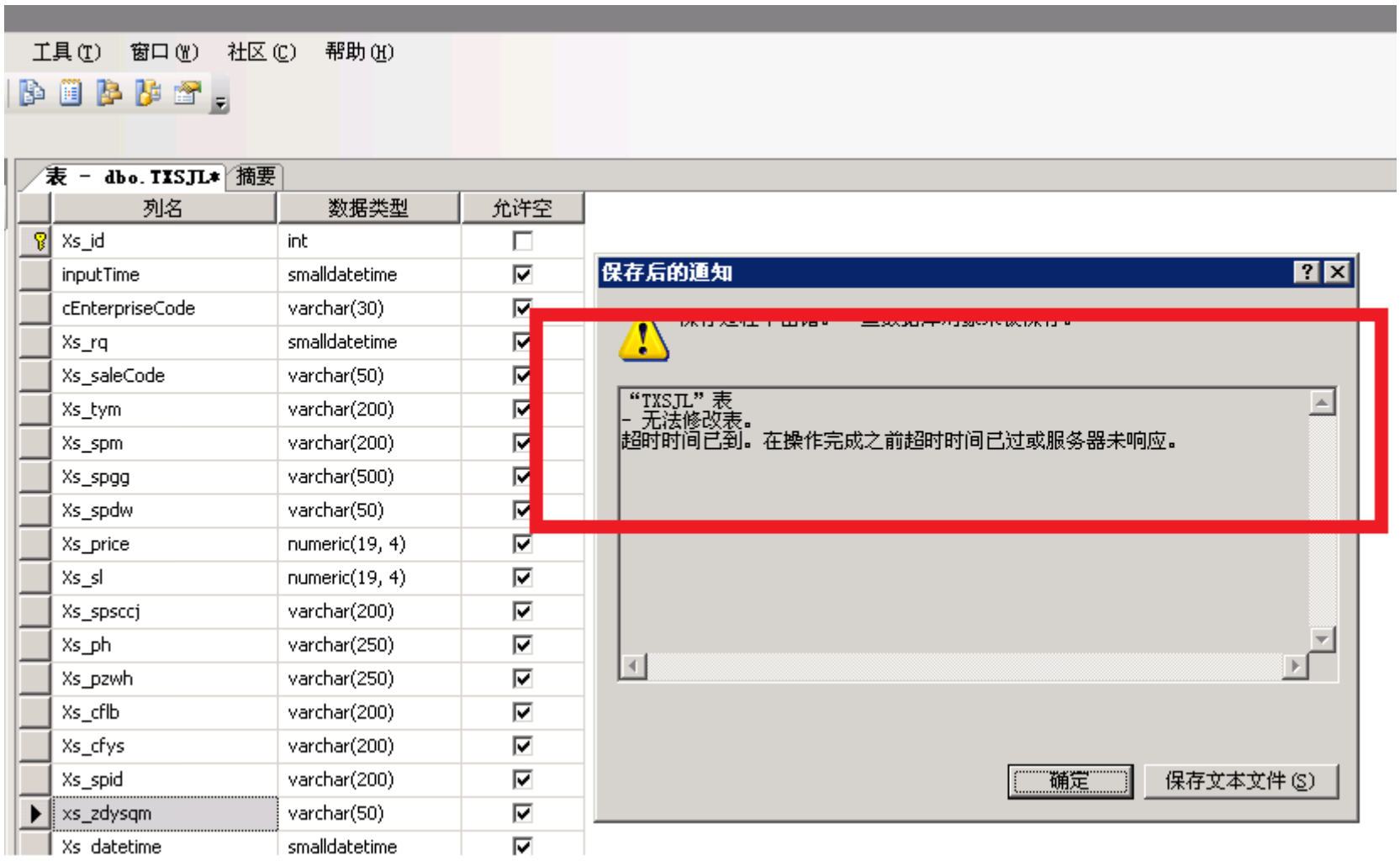
那再讲讲，解决问题的方法步骤：

1：首先优化数据库、因为程序也很复杂一时也看不过来也不敢乱改，**先从数据库字段类型优化开始入手**会好很多。

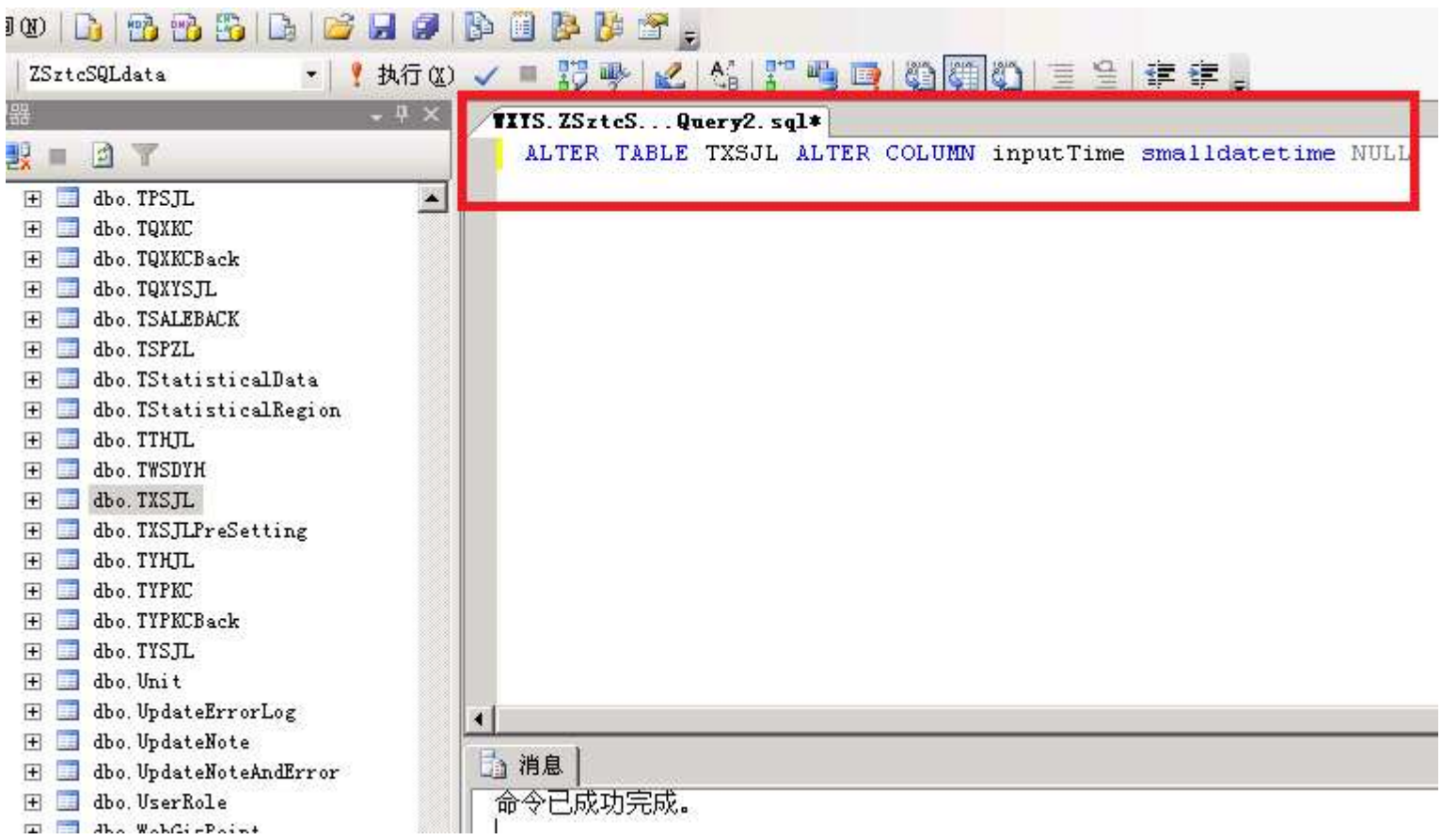
先把数据库里的 datetime 都修改为 smalldatetime，数据库变小了几百 M 很有成就感，最起码磁盘的读取压力减少不少吧。由于数据库数据有上千万条，无法用管理工具修改结构，只能用新建查询执行 SQL 命令才可以。

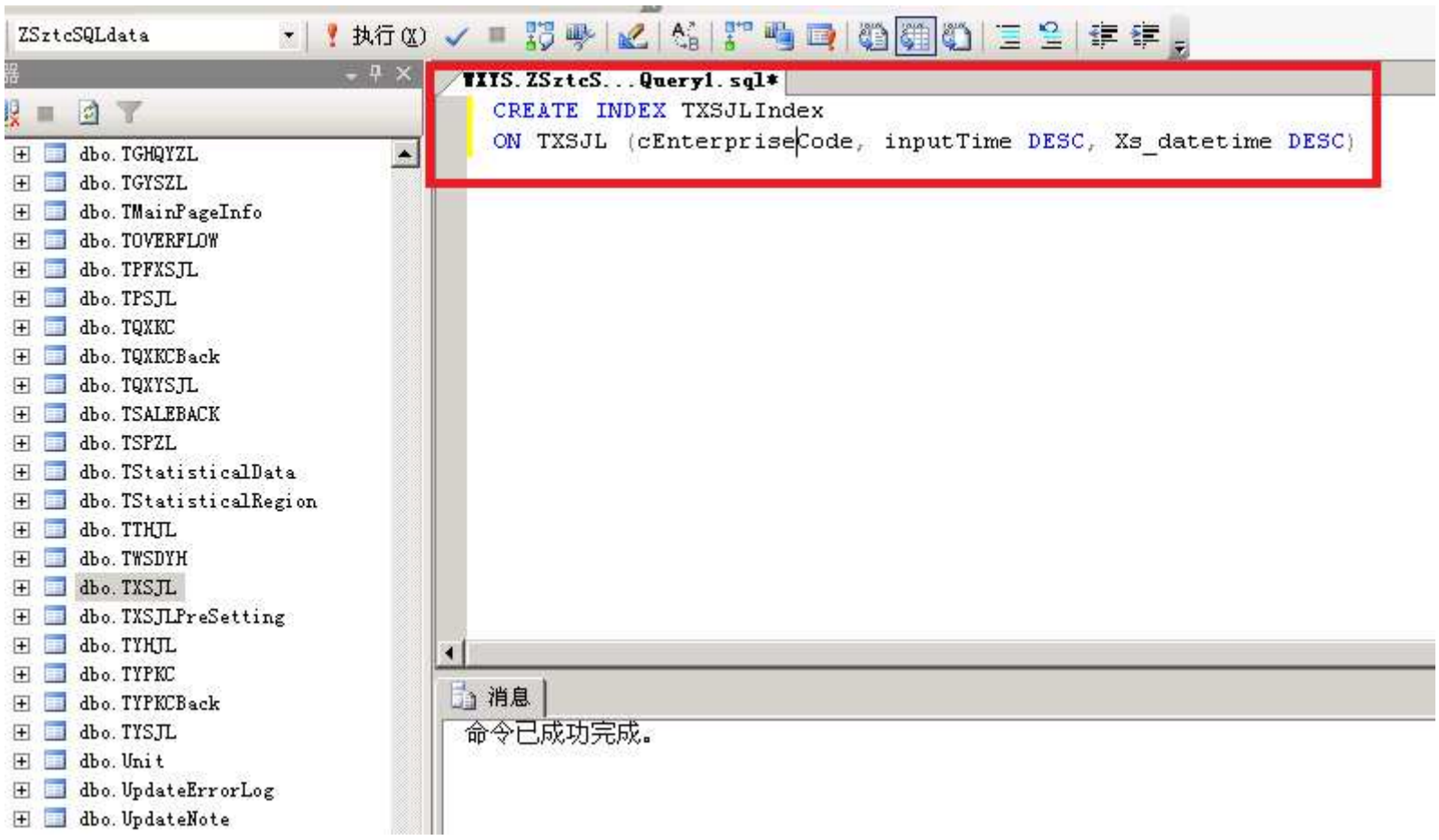


会有如下超时现象会发生。



那我们只能用执行查询的方式对表结构进行调整了，每次执行一个 SQL 指令大概需要 10 分钟时间才能顺利执行好，数据量实在是太大了。

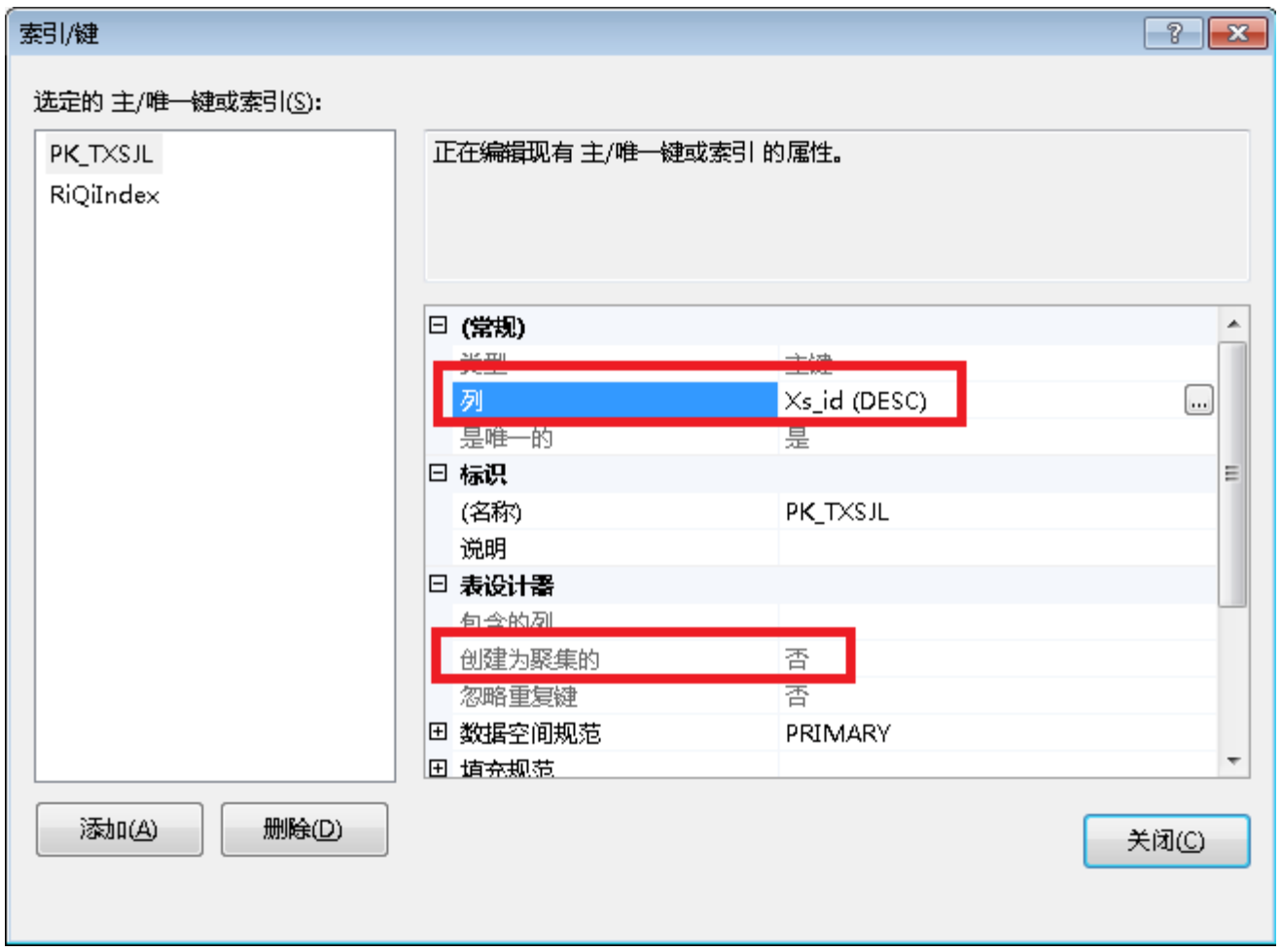




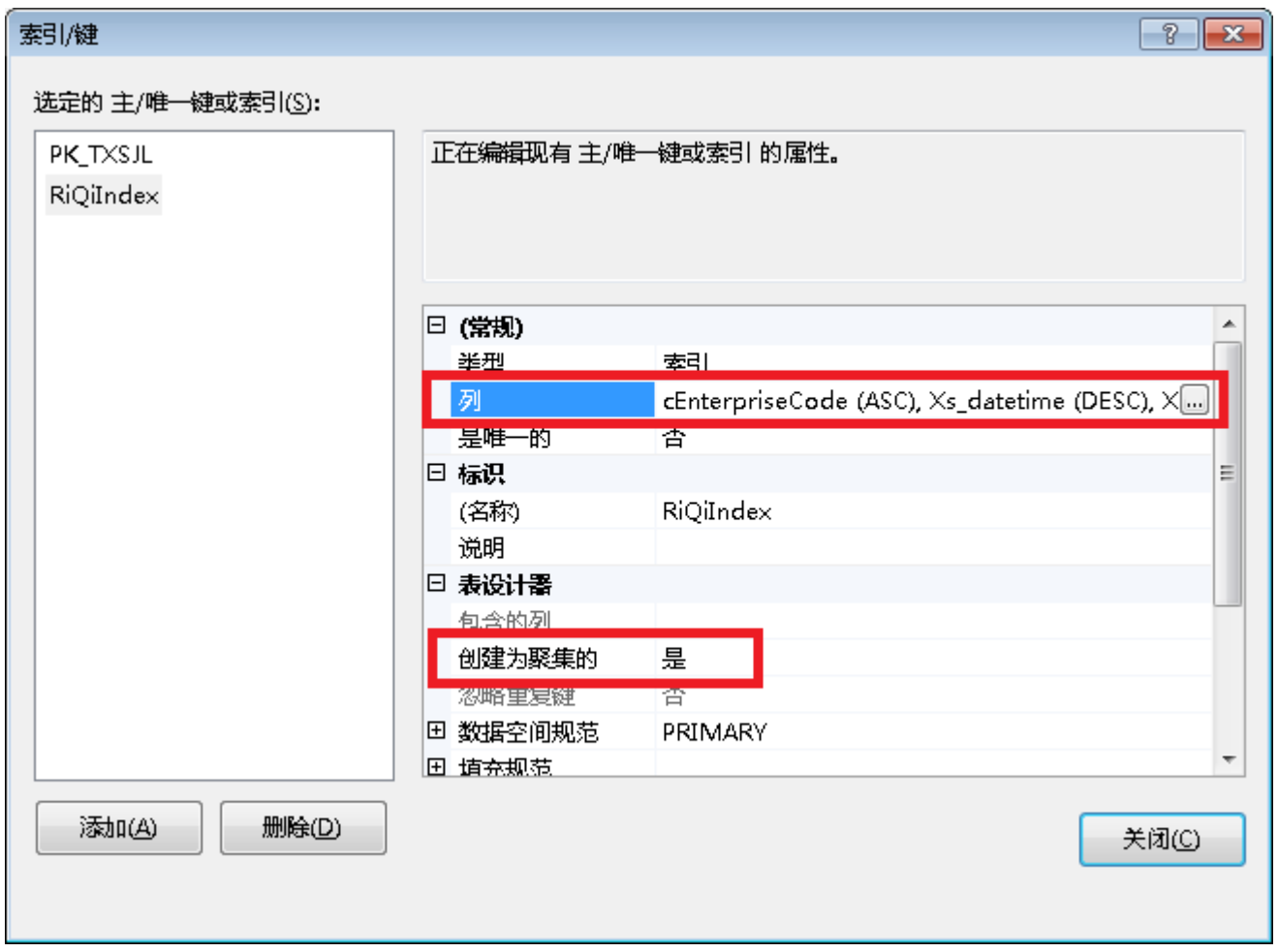
2：接着再优化，**数据库所引**，原先的所引很乱可以理解为是乱来的所以我全部干掉重新进行了组织。

把多余的所引先通通干掉，然后重新建立所引，因为记录数太庞大了，有多余的所引会使数据库变大很庞大，给他先减轻减轻体重。

把主键设置为倒序的、非聚集的，这样的好处是可以把最新的数据排序在最前面。

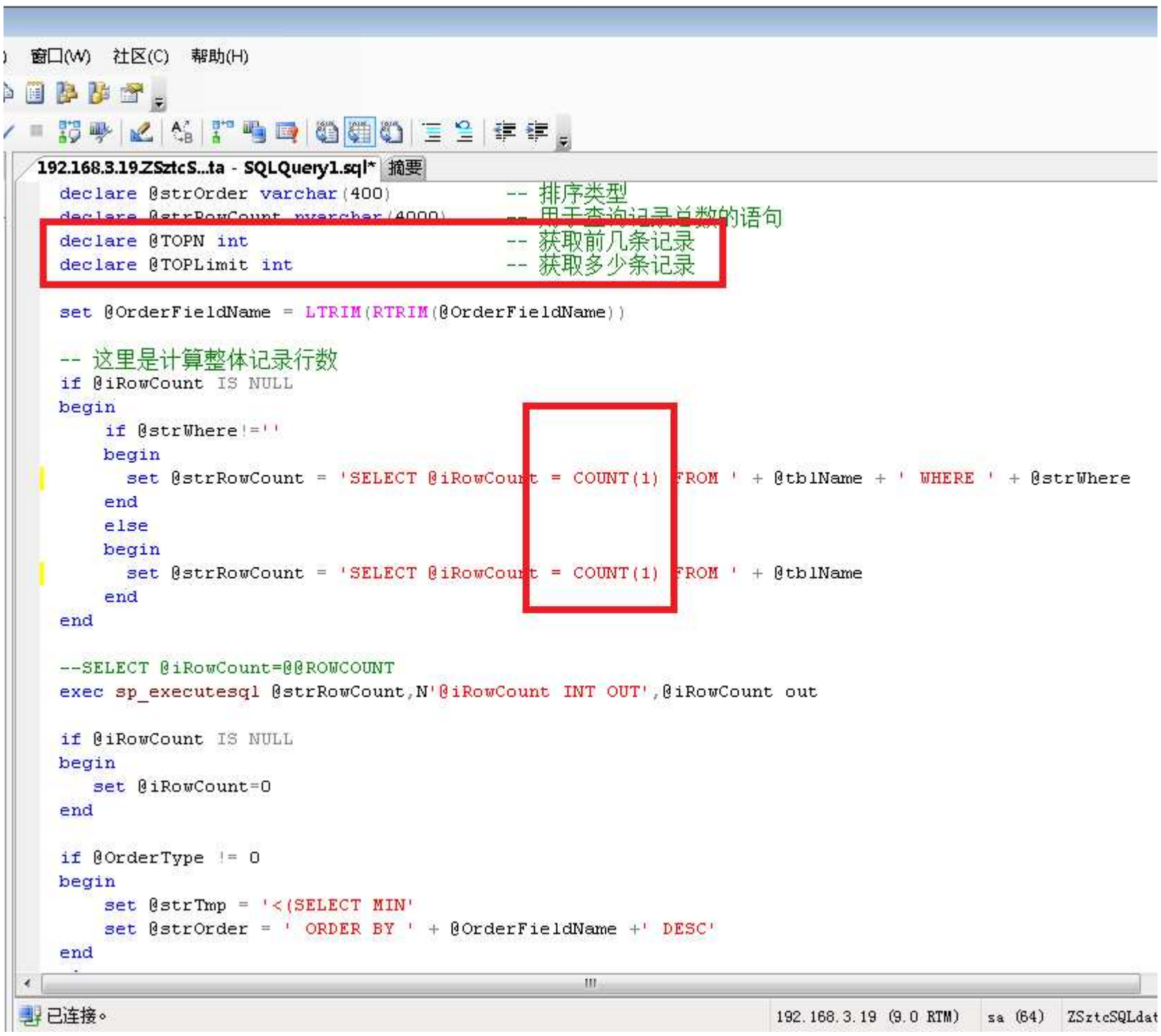


把主要查询的条件设置为所引，Group By 的放第一个位置然后设置为聚集所引，这样的好处时查询时会快很多很多，普通所以没这个效率高，数据实在是太庞大了，超过了 1000 万条数据后，对比一下还是很明显的，都能感觉得到。



完成以上 2 个步骤后分页速度快了很多最起码没死机现象了， 还有一点遗憾是当数据量大时最后一页的分页速度还是有些慢，有些难以忍受的感觉，但是最起码不会死机了。

3：接着重点优化，**数据库分页的存储过程**，最后一页难以忍受的问题先解决一下。分页是用了 SELECT TOP N 的反转的方式，我把最后一页到底获取多少条记录准确数字计算出来，适当的修改了一下最后一页慢得死去活来的问题，得到了适当的环节，虽然没能彻底解决也速度明显快了一些，由于写的这个分页程序也有些复杂，我也不敢乱动，就把问题解决好就完事大吉的目的了，不去惹更多的麻烦了。

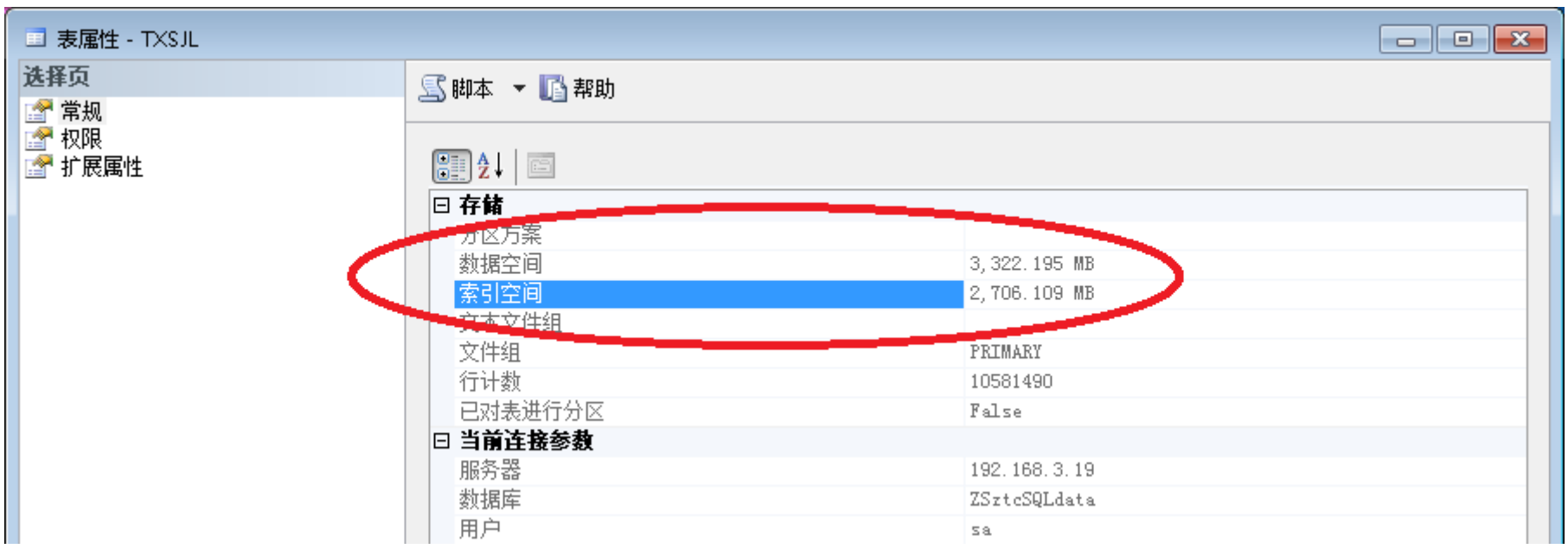

窗口(W) 社区(C) 帮助(H)
192.168.3.19.ZSztcS...ta - SQLQuery1.sql* 摘要
declare @strOrder varchar(400) -- 排序类型
declare @strRowCount nvarchar(4000) -- 用于查询记录总数的语句
declare @TOPN int -- 获取前几条记录
declare @TOPLimit int -- 获取多少条记录
set @OrderFieldName = LTRIM(RTRIM(@OrderFieldName))
-- 这里是计算整体记录行数
if @iRowCount IS NULL
begin
if @strWhere!=''
begin
set @strRowCount = 'SELECT @iRowCount = COUNT(1) FROM ' + @tblName + ' WHERE ' + @strWhere
end
else
begin
set @strRowCount = 'SELECT @iRowCount = COUNT(1) FROM ' + @tblName
end
end
--SELECT @iRowCount=@@ROWCOUNT
exec sp_executesql @strRowCount,N'@iRowCount INT OUT',@iRowCount out
if @iRowCount IS NULL
begin
set @iRowCount=0
end
if @OrderType != 0
begin
set @strTmp = '<(SELECT MIN'
set @strOrder = ' ORDER BY ' + @OrderFieldName +' DESC'
end
已连接。
192.168.3.19 (9.0 RTM)
sa (64)
ZSztcSQLdat

192.168.3.19.ZSztcS...ta - SQLQuery1.sql* 摘要

```
-- 获取几条数据？ 吉日嘎拉 2010-11-02 更新
set @TOPN = @iRowCount - @PageSize * (@PageIndex - 1)
if @TOPN > @PageSize
begin
    set @TOPN = @PageSize
end

set @TOPLimit = @PageSize * (@PageIndex -1)
if @TOPLimit > @iRowCount
begin
    set @TOPLimit = @iRowCount
end

set @strSQL = 'SELECT TOP ' + STR(@TOPN) + ' ' + @SelectFieldName + ' FROM '
    + @tblName + ' WHERE ' + @OrderFieldName + @strTmp + '('
    + RIGHT(@OrderFieldName, LEN(@OrderFieldName) - CHARINDEX('.', @OrderFieldName)) + ') FROM (SELECT TOP ' +
    + ' ' + @OrderFieldName + ' FROM ' + @tblName  + @strOrder + ') AS tblTmp)'
    + @strOrder

if @strWhere != ''
    set @strSQL = 'SELECT TOP ' + STR(@TOPN) + ' ' + @SelectFieldName + ' FROM '
        + @tblName + ' WHERE ' + @OrderFieldName + @strTmp + '('
        + RIGHT(@OrderFieldName, LEN(@OrderFieldName) - CHARINDEX('.',@OrderFieldName)) + ') FROM (SELECT TOP '
        + ' ' + @OrderFieldName + ' FROM ' + @tblName + ' WHERE ' + @strWhere + ' '
        + @strOrder + ') AS tblTmp) AND ' + @strWhere + ' ' + @strOrder

if @PageIndex = 1
begin
    set @strTmp = ''
    if @strWhere != ''
        set @strTmp = ' WHERE ' + @strWhere

    set @strSQL = 'SELECT TOP ' + STR(@TOPN) + ' ' + @SelectFieldName + ' FROM '
```

4：对比一下数据库结构优化后的前后如下图

所引优化前所引占用空间 2706.109M



所引优化后所引占用空间 520.805M

我想就这么一个 1000w 条记录的表光所引就优化了 2200M 空间，就单单这个也提高不少性能了。

5：接着重点优化，**程序代码部分了**，其实代码优化是在所引优化之前的，因为先读懂了代码、读懂了业务逻辑才好优化所引，这边文章写着写着顺序有些颠倒了，大家心里有数就可以了，我还是按照我的思路继续写吧。

在上图的企业编号、企业名称等，在程序里都进行了 LIKE 处理，当数据库记录超过 1000 万条时，对字符进行 Like 操作，那真是会要命的，毕竟那么多数据都进行一次匹配，虽然电脑的运算速度很快，但是上千万条记录，这么被计算过一下，能快到哪里去啊？

改进方法：

A: 输入企业编号、企业名称修改为模糊查询，能明确定位一个药店的名称。

B: 若已经获得企业编号了，不再匹配企业名称，而且企业编号用 = 来判断，并把企业编号进行所引。

海量数据库分页优化总结：

折腾了接近 1 周左右，终于把这个 1 千多万条记录的数据表给优化好了，难题也解决好了**虽然不太科学也不专业也缺少理论依据、试验数据、图表对比、性能调试工具等等，但是还好把问题都解决好了，老鼠抓到了就是好猫咪了哈哈**。

数据库进行了彻底的翻天覆地的优化、程序代码也进行了彻底的翻天覆地的优化后，分页速度飞快了。每页显示 16 条、记录数 1087292 条、分页数为 67956 页，每页分页速度都完全在 3 秒内，最后一页也不会死机了，也蛮快的足够可以忍受了。

等有空时，再把最后一页分页速度慢的问题再深入解决一下，先不去惹麻烦了稍微休息一下再说。

优化的每个动作需要 10 分钟左右才会执行好，若做错一次基本上就代表半个小时白忙乎了，还需要删除掉，再重新执行修正过的 SQL 语句，所以一天下来优化的成果并不会非常明显、需要几天时间才能优化好。

将权限管理、工作流管理做到我能力的极致，一个人只能做好那么很少的几件事情。