相比在SQL Server 2000 中使用的分页方式，在SQL Server 2005中使用新的语法ROW_NUMBER()来分页效率要高出很多，但是很多人在使用ROW_NUMBER()这种分页方式时，使用的方法并不正确，以下列出不正确的和正确的做法并做简单分析：

首先假设我们已经创建了如下的表和索引并初始化了100万条数据：

```
CREATE TABLE [dbo].[Users]
(
    [ID] [int] IDENTITY(1,1) NOT NULL,
    [Name] [varchar](50) NULL,
    [Test] [nchar](10) NULL,
    CONSTRAINT [PK_Users] PRIMARY KEY CLUSTERED
    (
        [ID] ASC
    ) ON [PRIMARY]
) ON [PRIMARY]

CREATE UNIQUE NONCLUSTERED INDEX [Inx_Name] ON [dbo].[Users]
(
    [Name] ASC
) ON [PRIMARY]


DECLARE @index INT
SET @index=0
WHILE @index<1000000
BEGIN
    INSERT INTO [dbo].[Users]([Name],[Test]) values(@index,'walkingp')
    SET @index = @index + 1
END
```

不正确的使用方式(查出所有数据后再排序)：

```
SELECT ID,Name,Test
FROM (SELECT ROW_NUMBER() OVER(ORDER BY Name) AS RowNum,*
      FROM dbo.Users
      ) AS T
WHERE RowNum BETWEEN 5000 AND 5100
```

正确的使用方式如下(查出主键进行排序过滤，然后使用过滤后的主键来查找数据)：

```
SELECT A.ID,A.Name,A.Test
FROM dbo.Users AS A
INNER JOIN
(SELECT RowNum,ID
```

```
 FROM (SELECT ROW_NUMBER() OVER(ORDER BY Name) AS RowNum,ID
       FROM dbo.Users) AS T
 WHERE RowNum BETWEEN 4000 AND 4100) as B
ON A.ID = B.ID
ORDER BY B.RowNum
```

以下具体分析：

```
SET STATISTICS IO ON
SET STATISTICS TIME ON
GO

PRINT 'Error.5000-------------------------'
DECLARE @time DATETIME
DECLARE @ms INT
SET @time= GETDATE()

SELECT ID,Name,Test
FROM (SELECT ROW_NUMBER() OVER(ORDER BY Name) AS RowNum,*
      FROM dbo.Users) AS T
WHERE RowNum BETWEEN 5000 AND 5100

SET @ms=DATEDIFF(ms,@time,GETDATE())
PRINT @ms--毫秒数
GO

PRINT 'Right.5000-------------------------'
DECLARE @time DATETIME
DECLARE @ms INT
SET @time= GETDATE()

SELECT A.ID,A.Name,A.Test
FROM dbo.Users AS A
INNER JOIN
(SELECT RowNum,ID
 FROM (SELECT ROW_NUMBER() OVER(ORDER BY Name) AS RowNum,ID
       FROM dbo.Users) AS T
 WHERE RowNum BETWEEN 5000 AND 5100) AS B
ON A.ID = B.ID
ORDER BY B.RowNum

SET @ms=DATEDIFF(ms,@time,GETDATE())
PRINT @ms--毫秒数
GO

PRINT 'Error.500000-------------------------'
```

```
DECLARE @time DATETIME
DECLARE @ms INT
SET @time= GETDATE()

SELECT ID,Name,Test
FROM (SELECT ROW_NUMBER() OVER(ORDER BY Name) AS RowNum,*
      FROM dbo.Users) AS T
WHERE RowNum BETWEEN 500000 AND 500100

SET @ms=DATEDIFF(ms,@time,GETDATE())
PRINT @ms--毫秒数
GO

PRINT 'Right.500000------------------------'
DECLARE @time DATETIME
DECLARE @ms INT
SET @time= GETDATE()

SELECT A.ID,A.Name,A.Test
FROM dbo.Users AS A
INNER JOIN
(SELECT RowNum,ID
 FROM (SELECT ROW_NUMBER() OVER(ORDER BY Name) AS RowNum,ID
       FROM dbo.Users) AS T
 WHERE RowNum BETWEEN 500000 AND 500100) AS B
ON A.ID = B.ID
ORDER BY B.RowNum

SET @ms=DATEDIFF(ms,@time,GETDATE())
PRINT @ms--毫秒数
GO

PRINT 'Error.900000------------------------'
DECLARE @time DATETIME
DECLARE @ms INT
SET @time= GETDATE()

SELECT ID,Name,Test
FROM (SELECT ROW_NUMBER() OVER(ORDER BY Name) AS RowNum,*
      FROM dbo.Users) AS T
WHERE RowNum BETWEEN 900000 AND 900100

SET @ms=DATEDIFF(ms,@time,GETDATE())
PRINT @ms--毫秒数
GO

PRINT 'Right.900000------------------------'
```

```
DECLARE @time DATETIME
DECLARE @ms INT
SET @time= GETDATE()

SELECT A.ID,A.Name,A.Test
FROM dbo.Users AS A
INNER JOIN
(SELECT RowNum,ID
 FROM (SELECT ROW_NUMBER() OVER(ORDER BY Name) AS RowNum,ID
       FROM dbo.Users) AS T
 WHERE RowNum BETWEEN 900000 AND 900100) AS B
ON A.ID = B.ID
ORDER BY B.RowNum

SET @ms=DATEDIFF(ms,@time,GETDATE())
PRINT @ms--毫秒数
GO
```

以下是SQL的统计信息：

Error.5000----------

(101 row(s) affected)
Table 'Users'. Scan count 1, logical reads 15649, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.

SQL Server Execution Times:
  CPU time = 31 ms, elapsed time = 35 ms.

36

Right.5000----------

(101 row(s) affected)
Table 'Users'. Scan count 1, logical reads 325, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.

SQL Server Execution Times:
  CPU time = 0 ms, elapsed time = 7 ms.

6

Error.500000----------

(101 row(s) affected)
Table 'Users'. Scan count 1, logical reads 1532807, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.

SQL Server Execution Times:

CPU time = 1797 ms, elapsed time = 1789 ms.

1786

Right.500000----------

(101 row(s) affected)
Table 'Users'. Scan count 1, logical reads 1545, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.

SQL Server Execution Times:
CPU time = 453 ms, elapsed time = 454 ms.

453

Error.900000----------

(101 row(s) affected)
Table 'Users'. Scan count 1, logical reads 2758790, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.

SQL Server Execution Times:
CPU time = 3266 ms, elapsed time = 3280 ms.

3273

Right.900000----------

(101 row(s) affected)
Table 'Users'. Scan count 1, logical reads 2528, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.

SQL Server Execution Times:
CPU time = 750 ms, elapsed time = 748 ms.

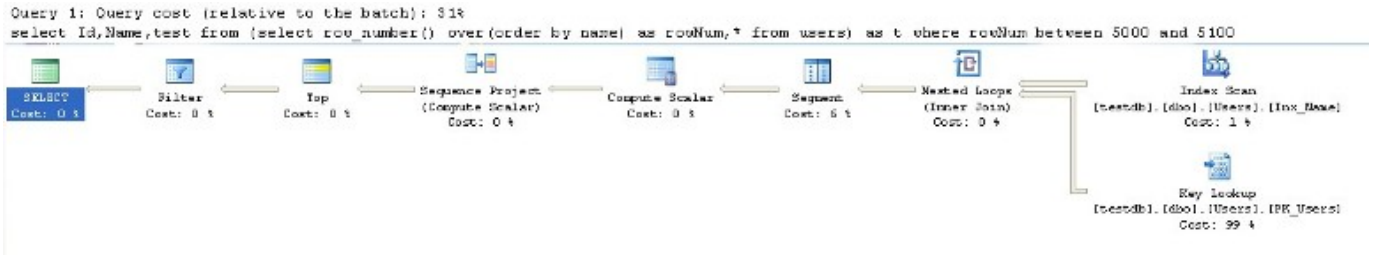
750

通过分析可以看出错误的使用方式逻辑读要比正确的使用方式的逻辑读大的多，而且页码越大读的越多，最终导致效率越来越差，这点也可以通过执行计划看出端倪。

以下是执行计划：

通过对比执行计划我们发现错误的使用方式在一开始就要读取聚集索引的数据分页中的数据，而正确的使用方式在一开始只是读取Inx_Name所引的所有数据分析，这在最后查出101条数据后才从聚集索引的数据分页中查找数据，因此效率要高的多，这种方式应该是创建此类SQL的一个通用原则。

错误的使用方式的执行计划：



正确的使用方式的执行计划：