과제3: 파일 구성 및 인덱스

다음의 스크립트를 실행하면 가상의 테이블에 데이터가 입력됩니다. 아무런 키도 설정하지 않고 입력하여 테이블을 생성하시오.

```
create table FileTest ( seqNo int identity(1,1),  randomNo int,    data char(900) )
declare @i int
set @i = 0
while @i < 8000
  begin
    insert into FileTest values(Round(rand()*100, 0), 'test data')
    set @i = @i + 1
  end
go
```

1) 위의 스크립트를 실행하면 테이블에 입력된 데이터는 대략 몇 페이지를 차지하는가? 행의 크기와 페이지의 크기를 고려해서 계산하시오. 각 페이지에는 페이지 관리에 필요한 약간의 공간이 필요하다.

```
DBCC DROPCLEANBUFFERS
DBCC FREEPROCCACHE

set statistics io on
select * from FileTest
set statistics io off

한행의 크기 4 + 4 + 900 ≒ 915
헤더(96)를 제외한 페이지 크기 추정 => 8K(8196) - 96 = 8096
한 페이지에 저장할 수 있는 최대 레코드 수 = 8096 / 915 = 8.8 즉 8개
따라서 8000개의 행 / 8 = 1000 페이지를 의미함.

 exec sp_spaceused filetest

 9200 / 8 = 1150,  8056/8 = 1007
```

2) FileTest를 조회하여 반환되는 행의 순서를 확인하고 seqNo가 3의 배수인 경우를 삭제하시오. 그리고 다시 위의 스크립트의 입력 부분을 수정하여 2666개의 행을 추가하시오. 조회하여 반환되는 행의 순서를 확인하면 순서가 바뀐 것을 알 수 있다. 그 이유를 쓰시오.

```
select * from Filetest
go

delete from FileTest
where (seqNo % 3) = 0
go

declare @i int
set @i = 0
while @i < (8000/3)
```

```
  begin
    insert into FileTest values(@i, 'test data')
    set @i = @i + 1
  end
go

select * from Filetest
go
```

현재는 인덱스가 없어서 insert 시 할당된 페이지들의 빈 공간에 입력되기 때문에 조회 시 순서가 없음

3) set statistics io on; 〈sql문〉; set statistics io off;는 〈sql문〉의 실행 시 페이지 읽기 횟수를 계산한다. Select * from FileTest where RandomNo 〈 30의 논리적 (페이지) 읽기 수를 구하시오.

```
set statistics io on
Select * from FileTest where RandomNo 〈 30
set statistics io off
```

4) FileTest의 seqNo에 대해 unclustered index를 설정하시오. 다시 select문으로 전체 행들을 조회하면 순서가 어떻게 변화하는지 확인하고 그 이유를 쓰시오.

```
CREATE NONCLUSTERED INDEX IX_seqNo ON FileTest(seqNo);
go

set statistics io on
select * from FileTest
set statistics io off
go
```

unclustered index는 데이터 페이지에 영향을 주지 않고 힙 파일에 대해 트리 구조만 생성함.

5) FileTest의 RandomNo에 대해 clustered index를 설정하시오. 다시 select문으로 전체 행들을 조회하면 순서가 어떻게 변화하는지 확인하고 그 이유를 쓰시오.

```
CREATE CLUSTERED INDEX IX_seqNo1 ON FileTest(RandomNo);
go

select * from FileTest
go
```

clustered index는 힙 파일 구조를 제거하고 트리의 리프 노드들을 데이터 페이지로 사용하기 때문에 RandomNo 순으로 정렬한다.

6) Select * from FileTest where RandomNo 〈 30의 논리적 (페이지) 읽기 수를 구하시오. 위의 실행 결과와의 차이를 쓰시오.

```
DBCC DROPCLEANBUFFERS
DBCC FREEPROCCACHE

set statistics io on
set statistics time on
```

```
Select * from FileTest where RandomNo < 30

set statistics time off
set statistics io off
```

논리적 읽기가 1/4 정도로 감소.
클러스터드 인덱스를 사용한 검색이고 인덱스는 RandomNo를 키로 함.

7) Select * from FileTest where seqNo % 10 = 0 의 논리적 (페이지) 읽기 수를 구하시오. 위의 실행 결과와의 차이를 쓰시오.

```
DBCC DROPCLEANBUFFERS
DBCC FREEPROCCACHE

set statistics io on
set statistics time on

Select * from FileTest where seqNo % 10 = 0

set statistics time off
set statistics io off
```

논리적 읽기 수는 1005로 변경됨.
unclustered index를 사용하고 있으므로 파일 스캔으로 처리됨.
%1000으로 변경해도 동일한 결과를 보여준다.
실시간으로 평가가 되기 때문에 plan을 만들지 못할 것이다.

8) Select seqNo from FileTest where Random < 30 의 논리적 (페이지) 읽기 수를 구하시오. 위의 실행 결과와의 차이를 쓰시오.

```
DBCC DROPCLEANBUFFERS
DBCC FREEPROCCACHE

set statistics io on
set statistics time on

Select seqNo from FileTest where RandomNo < 30

set statistics time off
set statistics io off
```

index only plan이 사용되기 때문에 (6)과 논리적 읽기 수의 차이 발생.

9) 임의의 한 행의 입력에 대한 논리적 (페이지) 읽기 수를 구하고 이를 설명하시오.

```
DBCC DROPCLEANBUFFERS
DBCC FREEPROCCACHE

set statistics io on
set statistics time on

insert into FileTest values(Round(rand()*100, 0), 'test data')

set statistics time off
set statistics io off
```

인덱스가 두 개 설치되어 있는 상황이기 때문에 삽입이 발생하면 인덱스에 대한 조정이 필요함.