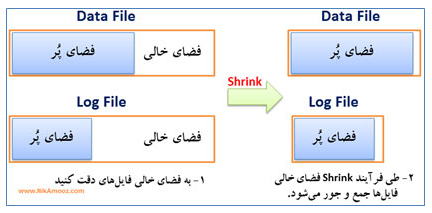
**بسمه تعالی**



برنامه نویسی بانک اطلاعاتی SQL Server

**نویسنده : محمد حسین فخرآوری**

Shrink در لغت به معني جمع شدن و يا چروک شدن مي‌باشد. با در نظر گرفتن همين مفهوم مي‌توان گفت Shrink کردن فرآيندي است که در آن فضاي Data File و Log File جمع و جور مي‌شود.



همانطور که در تصوير بالا مشاهده مي‌کنيد طي فرآيند دستور Shrink در SQL فضاي خالي فايل‌هاي بانک اطلاعاتي تا حد امکان از بين رفته و داده‌ها در يک قسمت جمع مي‌گردند.

جهت Shrink کردن بانک اطلاعاتي مي‌توان از دستور DBCC ShrinkDatabase استفاده نمود شکل کلي اين دستور به صورت زير مي‌باشد.

|  |
| --- |
| DBCC ShrinkDatabase  (  database\_name | database\_id | 0  [ , target\_percent ]  [ , { NOTRUNCATE | TRUNCATEONLY } ]  ) |

**پارامترهاي اين دستور به شرح زير مي‌باشد**

**۱ Database Name (**نام بانک اطلاعاتي که قرار است عمليات Shrink بر روي فايل‌هاي آن اتفاق بيافتد. لازم به ذکر است شما مي‌توانيد به جاي نام بانک اطلاعاتي از ID بانک اطلاعاتي هم به عنوان پارمتر جايگزين استفاده نماييد.

**۲ Target Percent (**اين پارامتر مشخص مي‌کند که چند درصد از فضاي خالي فايل مورد نظر پس از Shrink در دسترس باشد.

**۳(** پارامتر سوم شامل دو حالت زير است.

**TruncateOnly** در اين حالت چنانچه در انتهاي فايل مورد نظر فضاي خالي وجود داشته باشد اين فضاي خالي به سيستم عامل بازگشت داده مي‌شود. همچنين اگر TruncateOnly با Target Percent تواماً مورد استفاده قرار گيرد در اين صورت Target Percent ناديده گرفته مي‌شود. نکته مهمي که درباره TruncateOnly وجود دارد اين است که اگر اين پارامتر با دستور DBCC ShrinkDatabase مورد استفاده قرار گيرد تاثير آن بر Log File مي‌باشد و چنانچه شما خواهان تاثير عملکرد آن بر روي Data File باشيد بايد از دستور DBCC ShrinkFile استفاده نماييد.

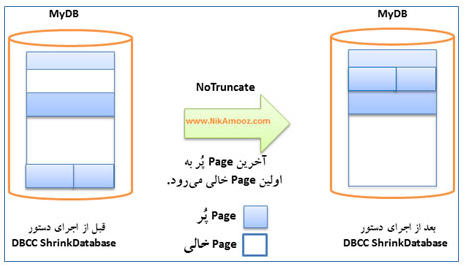
**NoTruncate**  عملکرد اين حالت صرفاً بر روي Data File بوده و طي آن آخرين فضاي پر Page پر در Data File به اولين فضاي خالي Page خالي منتقل مي‌شود. طي اين حالت Pageهاي Data File به بهترين نحو ممکن پر مي‌شود. اما اين موضوع باعث کاهش Performance بانک اطلاعاتي مي‌شود.

نکته مهمي که درباره NoTruncate وجود دارد اين است که تاثير اين پارامتر چه با دستور DBCC ShrinkFile و چه با دستور DBCC ShrinkDatabase صرفاً بر روي Data File مي‌باشد. همچنين اين در صورت استفاده از اين پارامتر هيچ فضاي خالي به سيستم عامل بازگشت داده نمي‌شود.

**مثال : دستور زير را در نظر بگيريد**

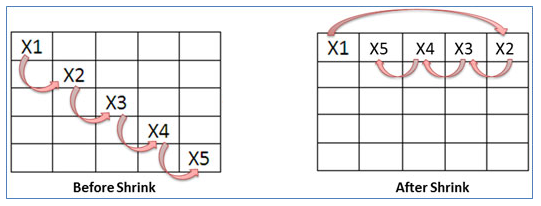
|  |
| --- |
| DBCC ShrinkDatabase(N'MyDB',NOTRUNCATE) |

تاثير اجراي اين دستور بر روي Data Fileهاي بانک اطلاعاتي بوده و طي آن جابجايي بين Pageهاي بانک اطلاعاتي رخ مي‌دهد. بدين صورت که آخرين Page پر به اولين Page خالي منتقل مي‌شود. تصوير زير اين موضوع را به درستي نمايش مي‌دهد.



اما اگر يادتان باشد در ابتداي مقاله اشاره شده که ShrinkDatabase به صورت NoTruncate کارايي بانک اطلاعاتي را پايين مي‌آورد. دليل اين موضوع اين است که در طي اين حالت با توجه به اينکه آخرين Page پُر به اولين Page خالي منتقل مي‌شود انديکس‌ها Fragment مي‌شوند.

اگر بخواهيم اين موضوع را دقيق‌تر بررسي کنيم بايد به تصوير زير دقت کنيد. قبل از انجام عمليات Shrink داده‌هاي ما X1 الي X5 به شکلي تقريباً منظم (مطابق آدرس منطقي) کنار هم قرار گرفته‌اند. پس از انجام عمليات Shrink مطابق تعريف ارائه شده براي حلت TruncateOnly آخرين فضاي پر به اولين فضاي خالي منتقل مي‌شود. در طي اين حالت چينش داده‌هاي ما کلاً عوض مي‌شود.که اين موضوع کارايي بانک اطلاعاتي را پايين مي‌آورد.

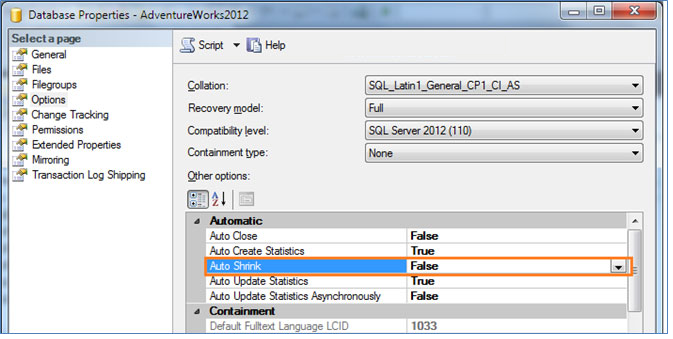


پس به طور خلاصه بايد گفت که Shrink  کردن Data File باعث بوجود آمدن Fragmentation در ايندکس‌ها و جداول مي‌شود که طي اين حالت آدرس منطقي و فيزيکي Pageها يکسان نخواهد بود و اين موضوع باعث مي‌شود که کوئري‌هاي ما IO بيشتري جهت واکشي Data داشته باشند.

**چند نکته مهم درباره دستور Shrink**

۱. دستور DBCC ShrinkFile جهت Shrink کردن يکي از فايل‌هاي بانک اطلاعاتي مورد استفاده قرار مي‌گردد. پارامترهاي آن مشابه به دستور DBCC ShrinkDatabase مي‌باشد. البته لازم به ذکر است اين دستور يک پارامتر اضافي هم دارد. (خارج از موضوع بحث مي‌باشد.) جهت کسب اطلاعات بيشتر در مورد اين دستور مي‌توانيد به اين لينک مراجعه کنيد.

۲. در محيط‌هاي عملياتي خصيصه Auto Shrink بانک اطلاعاتي را به هيچ عنوان True نکنيد.



خوب تا اينجا با مفهوم Shrink آشنا شديم در ادامه هدف‌مان اين است که وضعيت Fragmentation يک جدول قبل از انجام عمليات Shrink و پس از انجام عمليات Shrink بررسي نماييم.

**جهت انجام اينکار مراحل زير را به ترتيب دنبال نماييد.**

**۱) ايجاد بانک اطلاعاتي تستي :** طي اين مرحله وجود بانک اطلاعاتي بررسي شده و در صورتيکه بانک اطلاعاتي وجود داشته باشد حذف و پس از آن پروسه ايجاد بانک اطلاعاتي انجام مي‌شود.

|  |
| --- |
| USE master  GO  IF DB\_ID('Test\_Shrink') IS NOT NULL  DROP DATABASE Test\_Shrink  GO  CREATE DATABASE Test\_Shrink  GO |

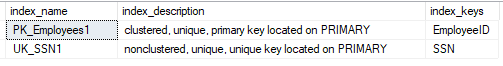
**۲) ايجاد دو جدول تستي :** طي اين مرحله وجود جداول بررسي شده و در صورتيکه جداول در بانک اطلاعاتي وجود داشته باشد حذف و پس از آن ايجاد مي‌گردند. به ازاي  جداول ايجاد شده دو Constraint در نظر گرفته شده است که يکي از آنها به عنوان Primary Key و ديگري به عنوان Unique Key در نظر گرفته شده است.

|  |
| --- |
| USE Test\_Shrink;  GO  IF OBJECT\_ID('Employees1') IS NOT NULL DROP TABLE Employees1;  GO  CREATE TABLE Employees1 (EmployeeID INT IDENTITY(1, 1),  SSN INT,  FirstName NCHAR(2000),  LastName NCHAR(2000),  CONSTRAINT PK\_Employees1 PRIMARY KEY(EmployeeID),  CONSTRAINT UK\_SSN1 UNIQUE(SSN));  GO  IF OBJECT\_ID('Employees2') IS NOT NULL DROP TABLE Employees2;  GO  CREATE TABLE Employees2 (EmployeeID INT IDENTITY(1, 1),  SSN INT,  FirstName NCHAR(2000),  LastName NCHAR(2000),  CONSTRAINT PK\_Employees2 PRIMARY KEY(EmployeeID),  CONSTRAINT UK\_SSN2 UNIQUE(SSN)); |

**نکته** : با توجه به اينکه هدف اين مثال بوجود آوردن حجم بالا براي جداول Data Typeهاي موجود در جداول NChar در نظر گرفته شده است.

۳) بررسي ايندکس‌هاي موجود در جدول : با استفاده از Stored Procedure سيستمي sp\_HelpIndex مي‌توانيد ايندکس‌هاي موجود در جداول را بررسي کنيد.

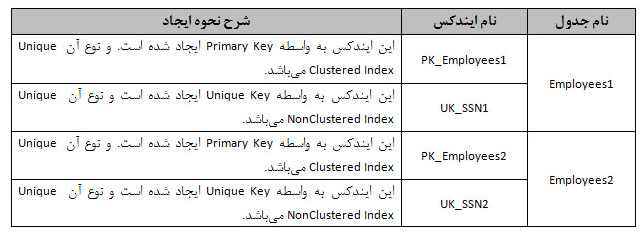
|  |
| --- |
| SP\_HELPINDEX Employees1  GO |



|  |
| --- |
| SP\_HELPINDEX Employees2  GO |



همانطور که در ليست ايندکس‌ها مشاهده مي‌نمايد جدول مورد نظر داراي دو ايندکس به شرح زير مي‌باشد.

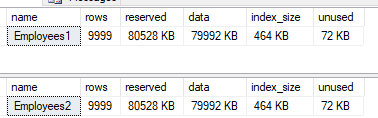


**۴) درج تعداد ۱۰۰۰۰ رکورد تستي در جداول :** توسط Scriptهاي زير مي‌توانيد با استفاده از يک حلقه While تعدادي رکورد تستي در جداول درج نماييد.

|  |
| --- |
| DECLARE @C INT=1  DECLARE @FirstName NVARCHAR(100)=''  DECLARE @LastName NVARCHAR(100)=''  WHILE @C <> 10000  BEGIN  SET @FirstName='FirstName' + CAST(@C AS NVARCHAR(10))  SET @LastName='LastName' + CAST(@C AS NVARCHAR(10))  INSERT INTO Employees1(SSN,FirstName,LastName) VALUES (@C,@FirstName,@LastName)  INSERT INTO Employees2(SSN,FirstName,LastName) VALUES (@C,@FirstName,@LastName)  SET @C+=1  END  SELECT \* FROM Employees1 |

۵) بررسي تعداد رکوردهاي درج شده :**با استفاده از Stored Procedure سيستمي sp\_SpaceUsed مي‌توانيد تعداد رکوردهاي موجود در جداول را بررسي کنيد.**

|  |
| --- |
| SP\_SPACEUSED Employees1  GO  SP\_SPACEUSED Employees2  GO |

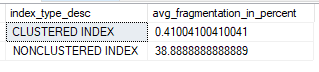


**۶) حذف جدول دوم :** با توجه به اينکه هدف مان شبيه‌سازي عمليات Shrink است جدول تستي دوم را حذف کنيد تا فضاي مربوط به آن در Data File بلا استفاده باقي مانده تا عمليات Shrink بتواند طي پروسه Shrink از آن استفاده نمايد.

|  |
| --- |
| DROP TABLE Employees2 |

**۷) بررسي وضعيت Fragmentation جدول و ايندکس هاي موجود در آن :** با استفاده از DMF (Dynamic Management Function) زير مي‌توانيد وضعيت Fragmentation ايندکس‌هاي موجود در جدول را بررسي کنيد.

|  |
| --- |
| SELECT index\_type\_desc, avg\_fragmentation\_in\_percent  FROM sys.dm\_db\_index\_physical\_stats(DB\_ID('Test\_Shrink'), OBJECT\_ID('Employees1'), NULL, NULL, 'Limited'); |



درصدهايي که در جدول زير مشاهده مي‌نماييد قبل از اجراي عمليات Shrink مي‌باشد.



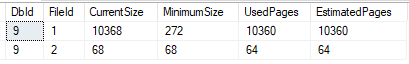
**۸) مشاهده تعداد IO جهت واکشي رکوردها :** با استفاده از دستور Set Statistics IO… مي‌توانيد تعداد IO لازم جهت واکشي کليه رکوردهاي جدول را مشاهده نماييد. لازم به ذکر است آمار ارائه شده براي IO قبل انجام عمليات Shrink مي‌باشد.

|  |
| --- |
| SET STATISTICS IO ON  GO  SELECT \* FROM Employees1  GO  SET STATISTICS IO OFF  GO |

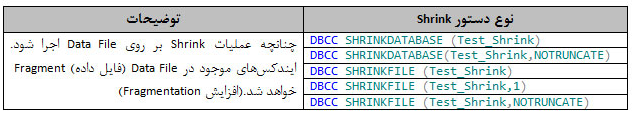


**۹) انجام عمليات Shrink :** عمليات Shrink بر روي Database انجام مي‌شود. نکته مهمي که در اين باره وجود دارد اين است که اگر عمليات Shrink بر روي تاثير خود را به Data File به شکل NoTruncate داشته باشد. اين موضوع باعث Fragment شدن جداول و ايندکس‌هاي شما خواهد شد.

|  |
| --- |
| DBCC SHRINKDATABASE (Test\_Shrink) |

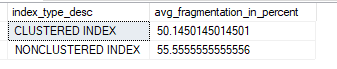


توجه داشته باشيد که اجراي هر کدام از دستورات زير به ضرر ايندکس‌ها مي‌باشد.



**۱۰) بررسي مجدد وضعيت Fragmentation جدول و ايندکس هاي موجود در آن :** با استفاده از DMF Dynamic Management Function زير مي‌توانيد وضعيت Fragmentation ايندکس‌هاي موجود در جدول را بررسي کنيد.

|  |
| --- |
| SELECT index\_type\_desc, avg\_fragmentation\_in\_percent  FROM sys.dm\_db\_index\_physical\_stats(DB\_ID('Test\_Shrink'), OBJECT\_ID('Employees1'), NULL, NULL, 'Limited'); |



درصدهايي که در جدول زير مشاهده مي‌نماييد بعد از اجراي عمليات Shrink مي‌باشد.



همانطور که در جدول بالا مشاهده مي‌کنيد عمليات Shrink تاثير خود را بر روي جداول و ايندکس‌هاي موجود در بانک اطلاعاتي گذاشته و باعث افزايش آمدن Fragmentation در آنها شده است.

**نکته مهم**

در صورتيکه Fragmentation ايندکس‌هاي شما به هر دليلي مانند Shrink کردن بانک اطلاعاتي و… رخ دهد بهتر است جهت افزايش کارايي بانک اطلاعاتي ايندکس‌هاي خود را بسته به شرايط Rebuild و يا Reorganize نماييد.