



عنوان مقاله: فشرده سازی داده در SQL Server

نویسنده مقاله: غلامحسین عبادی

تاریخ انتشار: اردیبهشت ۱۴۰۱

منبع: <https://nikamooz.com/data-compression-in-sql-server>

فشرده سازی داده در SQL Server

فشرده سازی یک جدول علاوه بر کاهش حجم دیسک مصرفی، به ما کمک می کند که IO کمتری اتفاق بیفتد. به عبارتی Physical Read و Logical Read کمتری اتفاق می افتد و تعداد صفحات کمتری از روی Disk و حافظه خوانده می شود و همچنین باعث کاهش ترافیک شبکه و نهایتاً کوئری شما با سرعت بالاتری اجرا خواهد شد. پس با فشرده سازی، داده ها در صفحات کمتری ذخیره می شوند و دستورات Select ما نیاز به خواندن صفحات کمتری از دیسک دارند. از آنجایی که Cost مربوط به Disk و Memory و CPU و Network می باشد، لذا ما با فشرده سازی می توانیم Cost مربوط به Disk و Memory را در هنگام Select به شدت کاهش دهیم. ولی اگر روی جداول فشرده بخواهیم عملیات Update انجام دهیم، CPU به شدت درگیر خواهد شد و Cost ما بالا خواهد رفت.

همینطور در جداول فشرده، هنگام عملیات Insert، خصوصاً عملیات Bulk، مدت زمان بیشتری صرف خواهد شد.

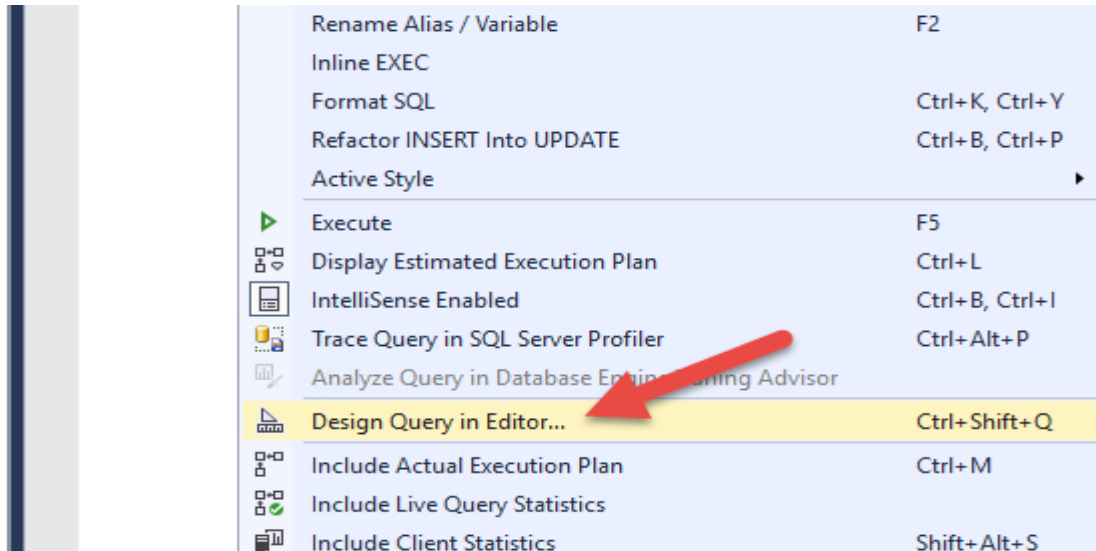
برای اطلاعات بیشتر به لینک زیر مراجعه کنید:

<https://docs.microsoft.com/en-us/sql/relational-databases/data-compression/data-compression?view=sql-server-ver15>

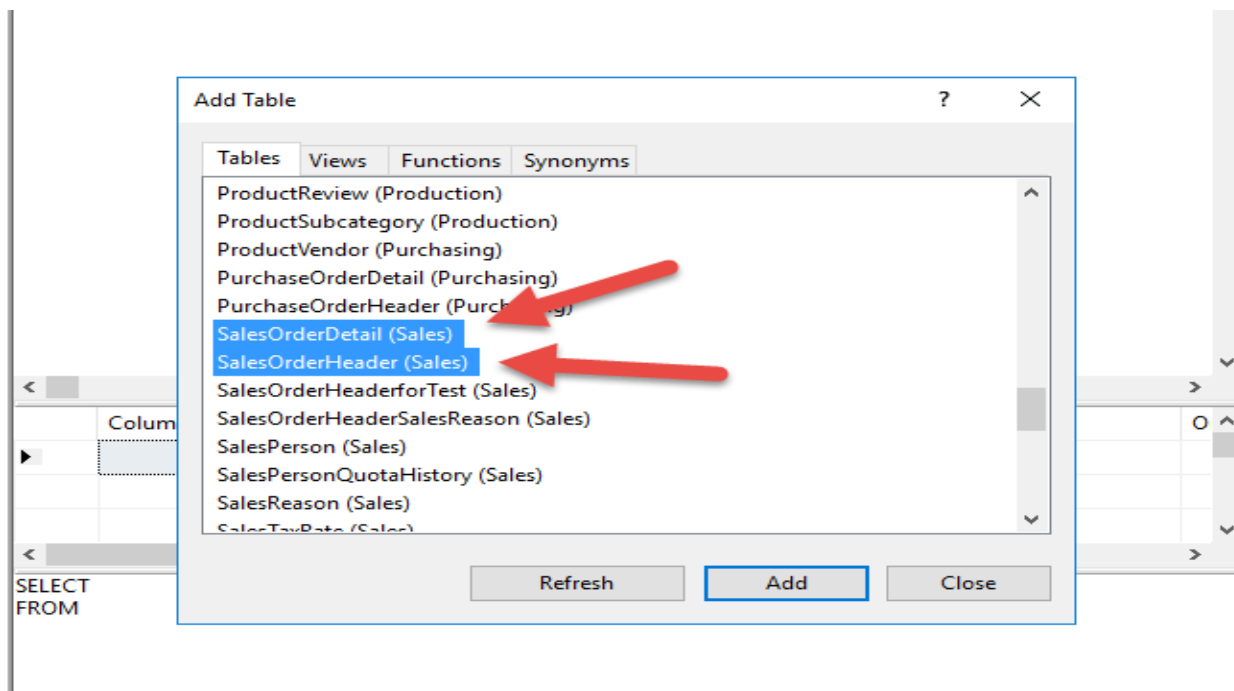
فشرده سازی را می توانیم در موارد زیر انجام دهیم:

- ۱ - A whole table that is stored as a heap.
- ۲ - A whole table that is stored as a clustered index.
- ۳ - A whole nonclustered index.
- ۴ - A whole indexed view.
- ۵ - For partitioned tables and indexes, you can configure the compression option for each partition, and the various partitions of an object do not have to have the same compression setting.

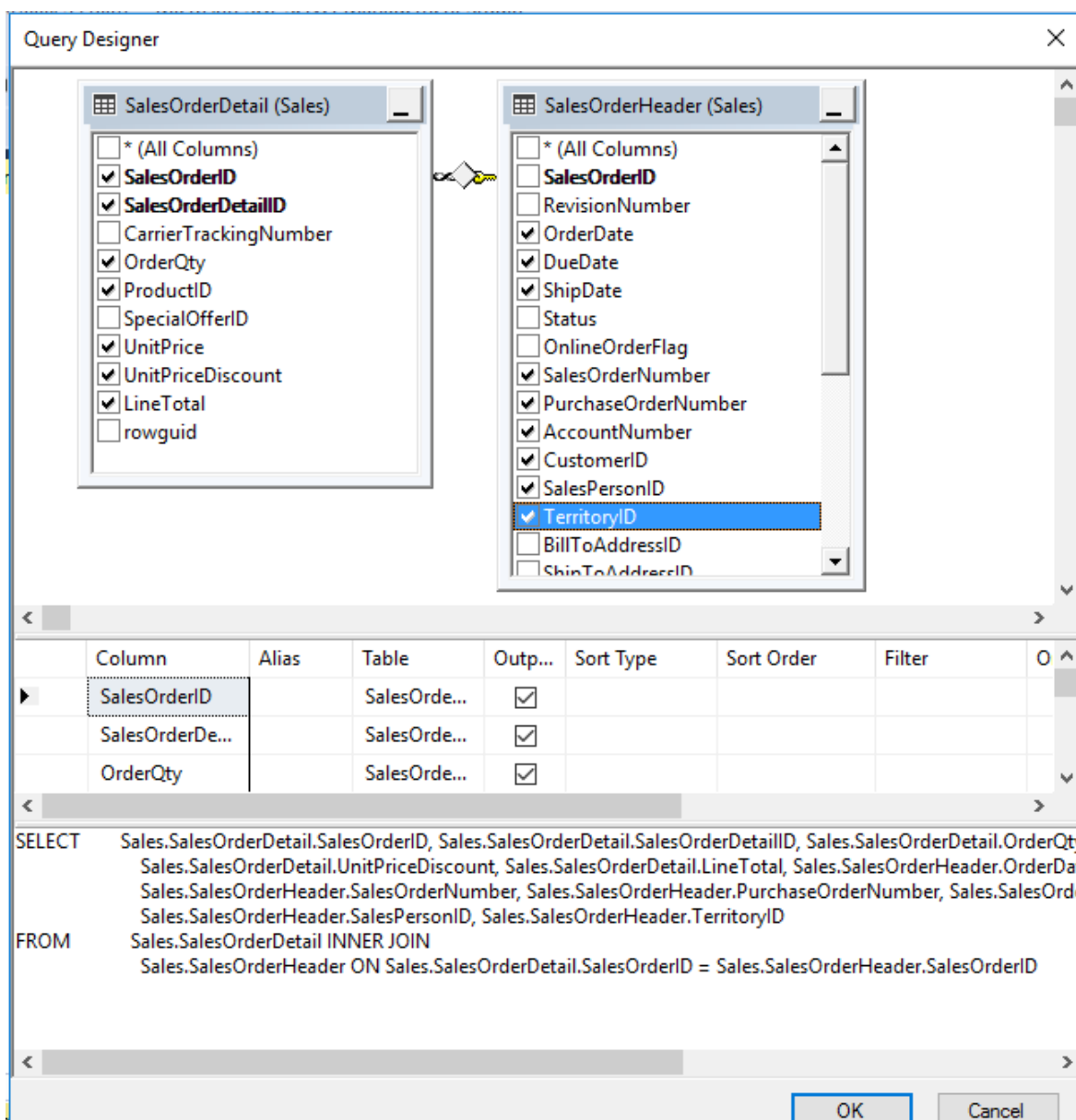
در این قسمت به یک مثال ساده می پردازیم. دیتابیس Adventureworks۲۰۱۹ را انتخاب نموده و سپس در محیط new Query کلیک راست نموده و بر روی گزینه Design Query in Editor مطابق شکل زیر کلیک کنید.



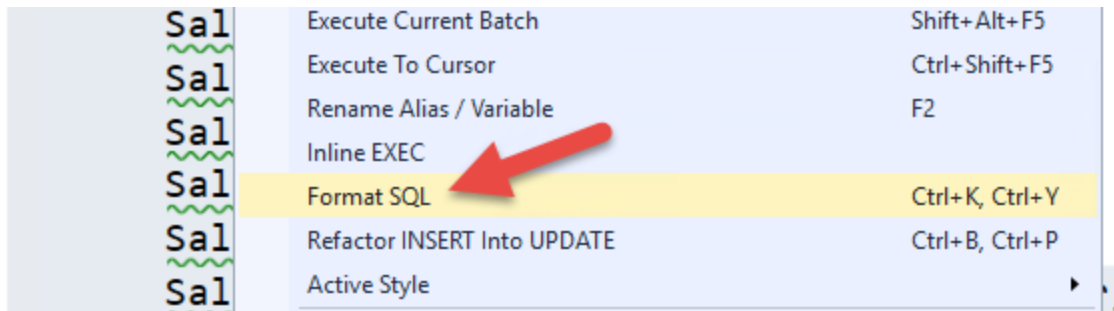
در پنجره ظاهر شده مطابق شکل زیر جداول SalesOrderDetail و SalesorderHeader را انتخاب کنید.



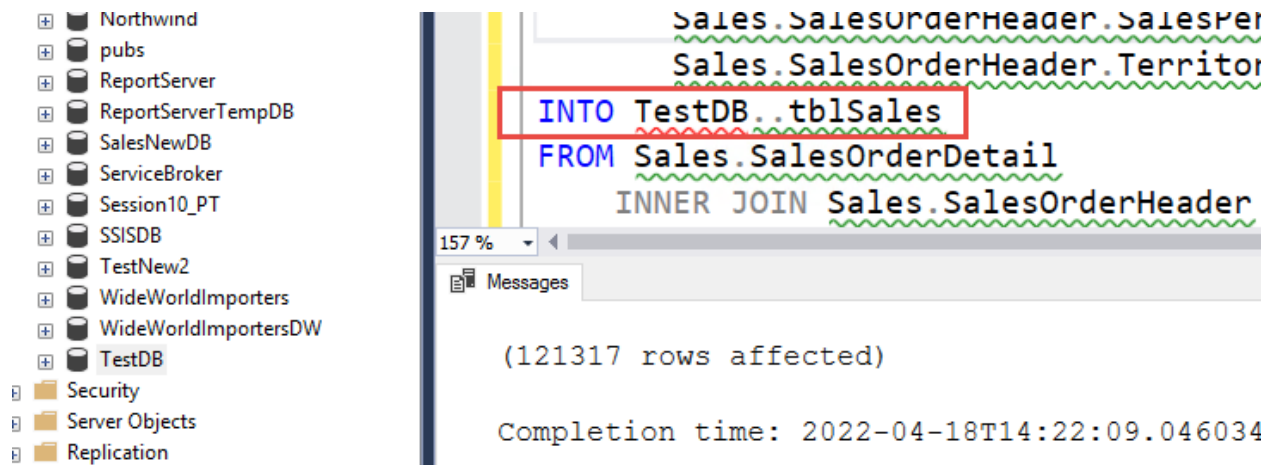
حال مطابق شکل زیر تعدادی از فیلدهای هر دو جدول را به دلخواه انتخاب کنید. در نهایت بر روی دکمه OK کلیک کنید تا کوئری مربوط به این انتخابها برای شما ایجاد شود.



تذکره: اگر بر روی سرور خود Redgate انتخاب نموده اید کفایست کل کوئری را انتخاب و بر روی آن کلیک راست و گزینه SQL Format و یا کلیدهای Ctrl+K, Ctrl+Y را بگیرید تا کدهای شما مطابق شکل زیر مرتبط شوند.



حال در این کوئری قبل از کلمه From از دستور INTO TestDB..tblSales استفاده کنید. (لازم به ذکر است که نام دیتابیس تستی ما TestDB و اسکیمای ما dbo و نام جدولی که بناست ایجاد شود tblSales نامیده می شود).



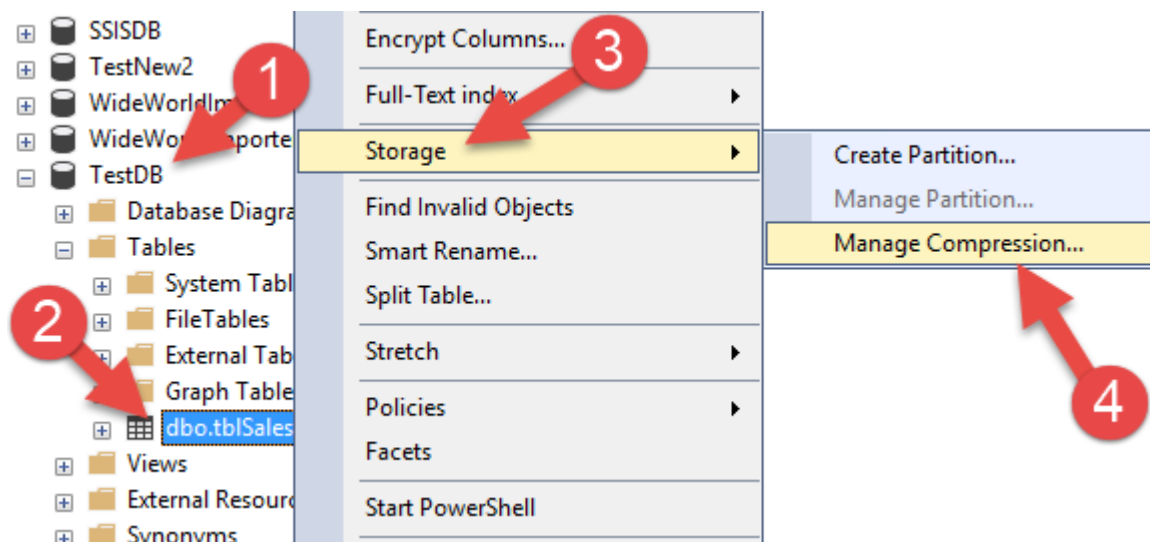
همانطور که در شکل فوق دیده می شود تعداد ۱۲۱۳۱۷ رکورد به جدول tblSales واقع در دیتابیس TestDB منتقل شد. لازم به ذکر است که شما می توانید با استفاده از دستور SP_Spaceused تعداد رکوردهای جدول tblSales را مشاهده نمایید.

تذکر: جدولی که به این روش ساخته می شود یک جدول Heap می باشد. شما می توانید با کوئری زیر جداول Heap را شناسایی کنید.

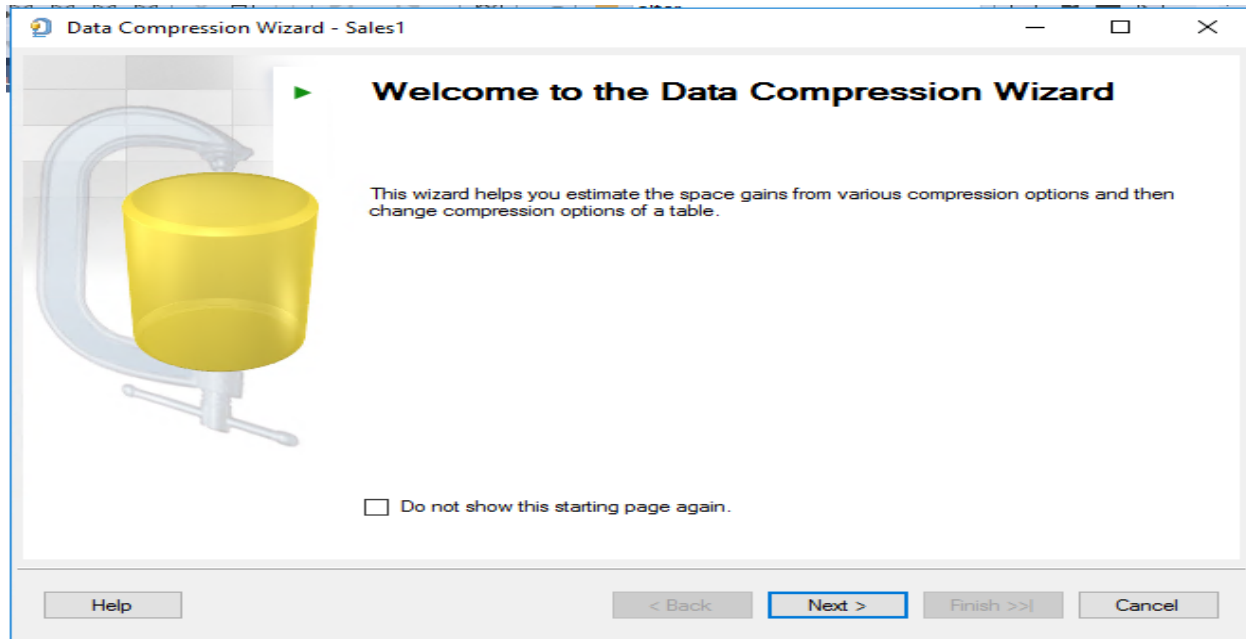
```
SELECT OBJECT_NAME(object_id),* FROM sys.indexes
WHERE type_desc='Heap'
```

(No column name)	object_id	name	index_id	type	type_desc	is_unique	data_spac
1	sysfiles1	8	NULL	0	HEAP	0	1
2	sqlagent_jobsteps_logs	149575571	NULL	0	HEAP	0	1
3	persistent_version_store	309576141	NULL	0	HEAP	0	1
4	persistent_version_store_long_term	325576198	NULL	0	HEAP	0	1
5	tblSales	597577167	NULL	0	HEAP	0	1

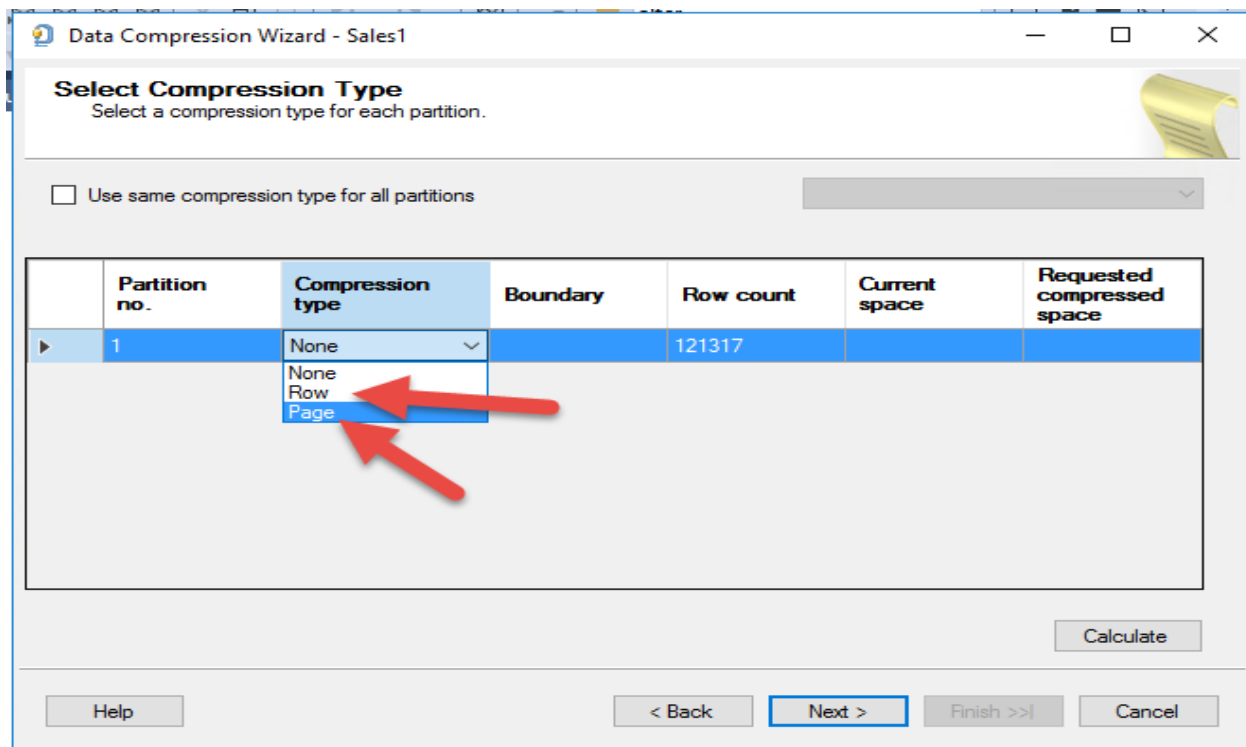
اگر بخواهیم جدولی را فشرده کنیم، هم می توانیم با استفاده از کوئری این کار را انجام دهیم و هم می توانیم به صورت Wizard ای این کار را انجام دهیم. اگر بخواهیم به صورت Wizard ای یک جدول را فشرده کنیم، کافیت روی جدول مورد نظرمان کلیک راست نموده و گزینه Storage و سپس گزینه Manage Compression را کلیک می نمایم.



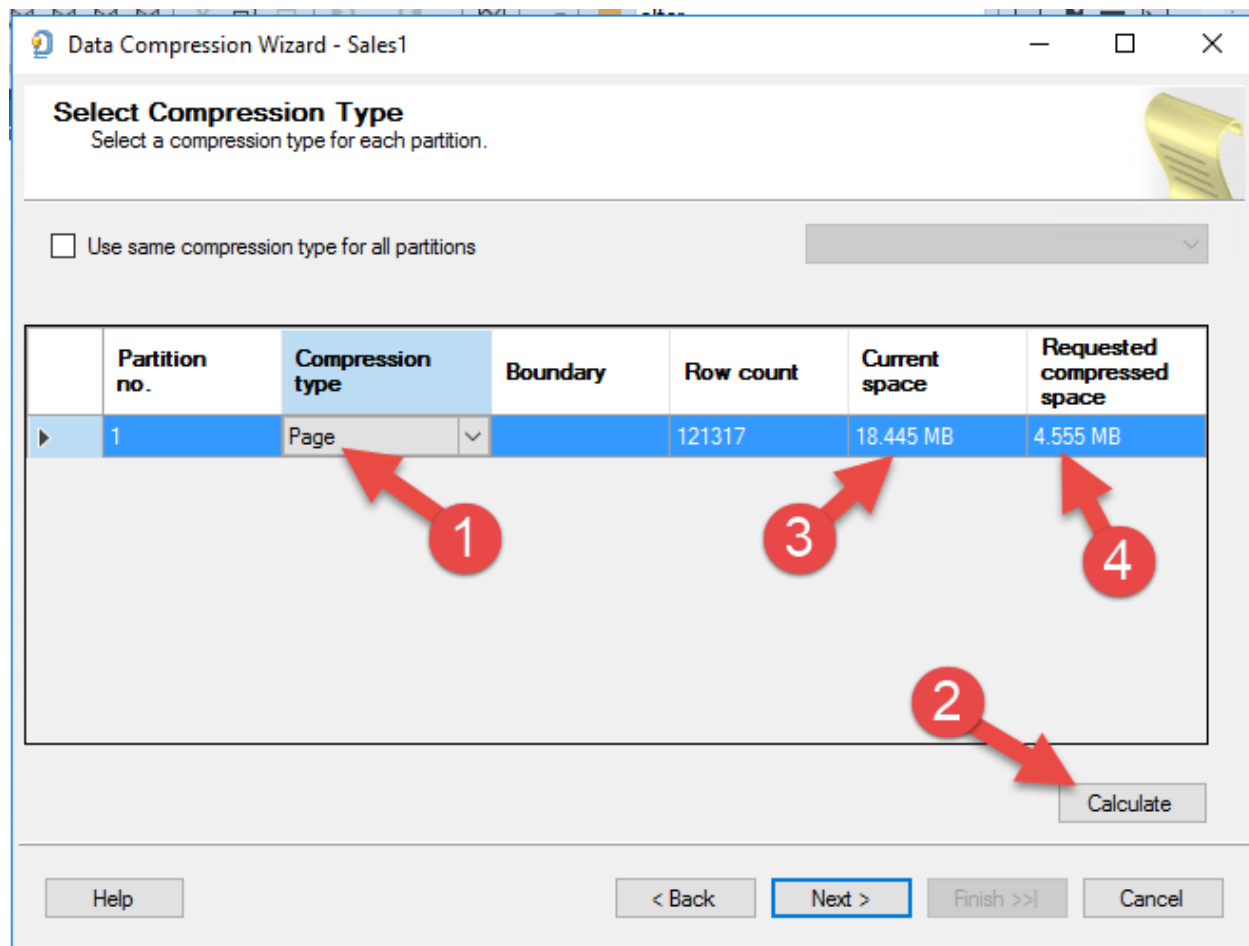
در این صورت پنجره ایی مطابق شکل زیر نمایان می شود(پنجره خوش آمد گویی). بر روی دکمه Next کلیک کنید تا به مرحله بعد بروید.



شما می توانید یک جدول را مطابق شکل زیر به دو حالت فشرده کنید. حالت Row Compression و حالت Data Compression (لازم به ذکر است که آیتم دیگر برای فشرده سازی به نام ColumnStore وجود دارد که در اینجا دیده نمی شود).



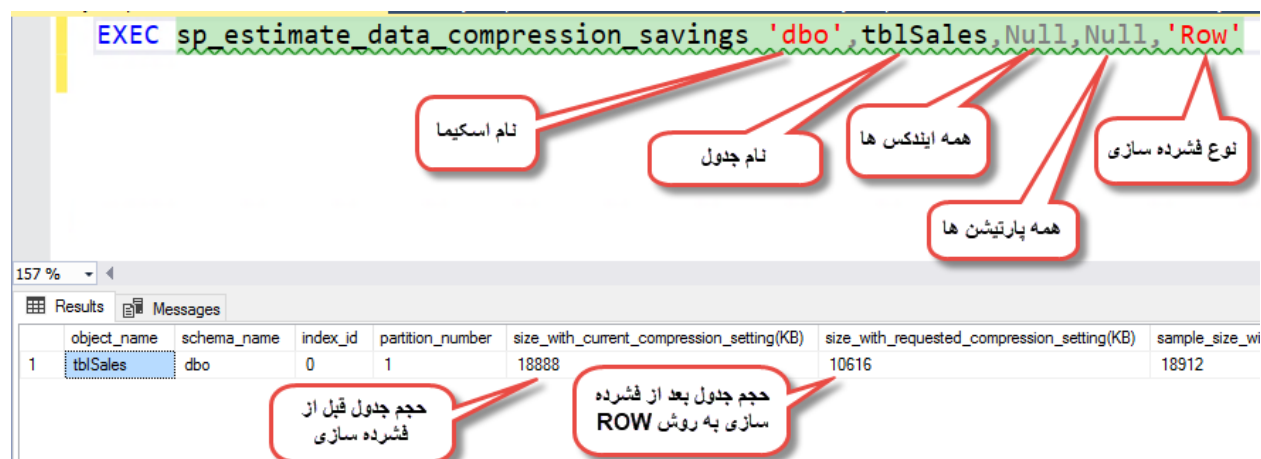
نوع فشرده سازی Row را انتخاب نموده و بر روی دکمه Calculate کلیک می نمایم. شما در پنجره زیرمیزان فشرده سازی بر اساس تکنیک Row Compression را مشاهده می کنید. در اثر این نوع فشرده سازی تا میزان تقریباً هشت مگابایت جدول شما فشرده خواهد شد. این میزان فشرده سازی همیشه ثابت نیست و به داده های شما بستگی دارد. حالت بعدی فشرده سازی بر اساس Page Compression می باشد. مطابق شکل زیر در اثر این نوع فشرده سازی ، حجم جدول شما از تقریباً هجده مگابایت به چهار مگابایت خواهد رسید(تقریباً یک چهارم).



همانطور که در دو شکل فوق می بینید ، فشرده سازی در سطح Page به مراتب کلانتر از فشرده سازی در سطح Row می باشد. در شکل فوق بر روی دکمه Next کلیک کرده تا داده های خود را فشرده کنیم.

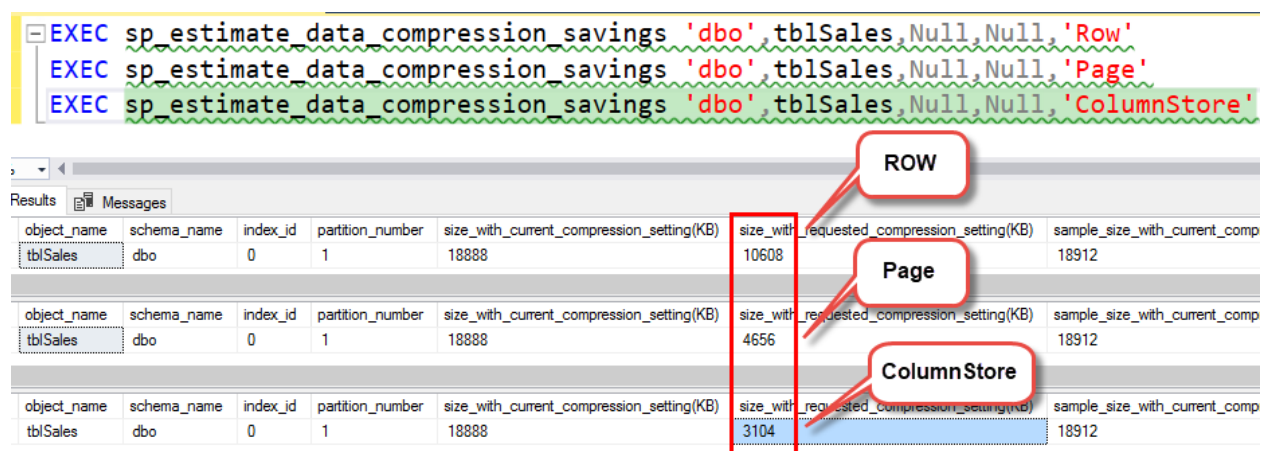
در اثر فشرده سازی دیتا چقدر فضا صرفه جویی خواهیم کرد؟

شما می توانید به کمک SP سیستمی به نام SP_estimate_data_compression_saving مطابق شکل زیر حجم مربوط به جدول را قبل و بعد از فشرده سازی را مشاهده نمایید. این SP مطابق شکل زیر دارای پنج پارامتر ورودی می باشد.



object_name	schema_name	index_id	partition_number	size_with_current_compression_setting(KB)	size_with_requested_compression_setting(KB)	sample_size_wi
tblSales	dbo	0	1	18888	10616	18912

حال اگر بخواهیم فشرده سازی به روش Row را با فشرده سازی به روش Page و فشرده سازی به روش ColumnStore مقایسه کنیم، مطابق شکل زیر خواهیم دید که در اثر فشرده سازی به روش Row حجم جدول ما از ۱۸MB به ۱۰MB تقلیل خواهد یافت و در روش Page حجم جدول از ۱۸MB به ۴MB و در روش ColumnStore حجم جدول از ۱۸MB به ۳MB تقلیل خواهد یافت.



object_name	schema_name	index_id	partition_number	size_with_current_compression_setting(KB)	size_with_requested_compression_setting(KB)	sample_size_with_current_comp
tblSales	dbo	0	1	18888	10608	18912
tblSales	dbo	0	1	18888	4656	18912
tblSales	dbo	0	1	18888	3104	18912

تذکر: ما با دستور ColumnStore مطابق کد زیر نمی توانیم داده ها را فشرده کنیم ولی مطابق شکل فوق می توانیم تخمین بزنیم که اگر روی جدول Column Store Index قرار دهیم به طور اتوماتیک داده های ما به چه میزان فشرده خواهد شد (کد زیر غلط است):

`ALTER TABLE [Sales].[SalesOrderDetail] Rebuild`

WITH(DATA_COMPRESSION = ColumnStore)

سوالی که اینجا به ذهن می رسد این است که آیا بهتر نیست که همه جداول خود را فشرده کنیم؟

نکته ایی که باید به آن توجه کنیم این است که فشرده سازی جداول باعث کاهش سرعت دستورات Insert و Delete و Update می شود، ولی به شدت باعث افزایش سرعت Select می شود. مخصوصا در جداول Fact مربوط به دیتابیس های Data Warehouse و پارتیشن های آرشیو جداول مربوط به جداول پارتیشن بندی شده (نه پارتیشن مثلا ماه جاری) و دیتابیس های آرشیو و سامانه هایی که از جنس Report هستند و به طور کلی جداولی که دارای Scan بالایی هستند و تقریبا Update ایی روی آنها صورت نمی گیرد ، گزینه مناسبی برای فشرده سازی هستند.

سوال: چه عملیاتی در اسکيوال سرور وجود دارد که کل جدول را Scan می کنند؟ به بعضی از آنها اشاره می کنیم.

- Cursor
- Order By
- Aggregate Function (چون Group By انجام می دهد)

لذا تا جایی که می توانید از موارد بالا که باعث کاهش Performance می شوند، استفاده نکنید.

تذکر مهم: وقتی جدولی را فشرده می کنید، این جدول Rebuild می شود و تمامی Dirty Page های مربوط به این جدول، روی دیسک، Persist می شوند. پس خود عملیات فشرده سازی هزینه بر می باشد.

وقتی جدولی را بر اساس Row Level Compression فشرده می کنید چه اتفاقی می افتد؟

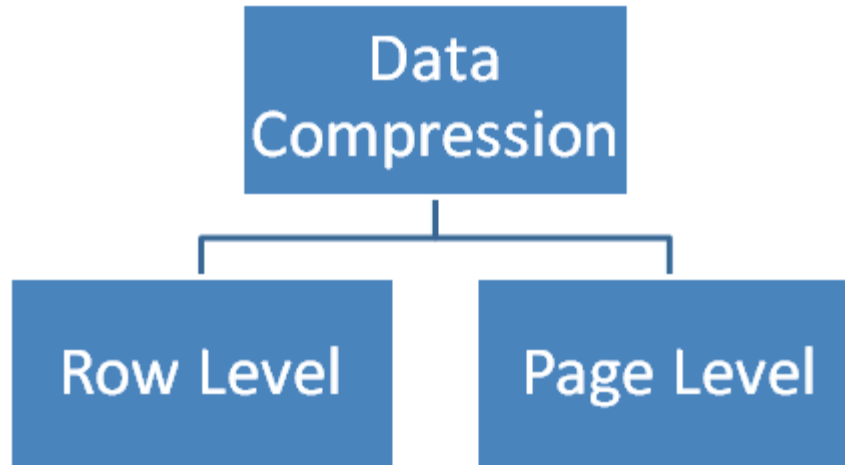
دوستان Data Type هایی که ما در اسکيوال سرور داریم می توانیم از یک نگاه آنها را به دو قسمت تقسیم بندی نماییم :

۱. Fix Length ها:

دیتا تایپ هایی که Fix Length هستند عبارتند از:

- عددی ها مثل: Tinyint , Smallint , int , bigint
- رشته ایی ها مثل: Char , NChar
- تاریخ مثل: SmallDateTime , DateTime , Date , Time
- سایر دیتا تایپ ها: مانند Uniqueidentifier

۲. Variable Length ها



Row Level Compression

فرض کنید در جدول SalesOrderDetail شما فیلدی به نام OrderQty دارید که دیتا تایپ مربوط به آن را از نوع Int انتخاب کرده اید. حال فرض کنید در آن مقدار عدد ۱۲ را وارد کنیم در این صورت چهاربایت فضا اشغال خواهد شد. حال اگر عدد ۲۱۴۷۰۰۰۰۰۰ هم وارد کنید باز هم چهار بایت فضا اشغال خواهد شد. وقتی ما جدولی را به روش ROW فشرده می کنیم، الگوریتم فشرده سازی آن به این صورت است که نگاه می کند میبینه که عدد ۱۲ رو می تونه توی یک فضای یک بایت هم جای بده، لذا به جای اینکه فضای چهار بایت به آن اختصاص بده، آن عدد ۱۲ را در فضای یک بایت جای می دهد. **به عبارت بهتر در این الگوریتم Fix Length ها را به Variable Length ها تغییر می یابند.** همچنین در الگوریتم ROW برای NULL ها فضا دیگر اشغال نمی شود و همچنین در این روش برای صفرها نیز فضا اشغال نمی شود. در نهایت این که الگوریتم فشرده سازی ROW بیشتر تمرکزش روی Data می باشد.

لازم به ذکر است که دیتا تایپ هایی همچون Tinyint و یا SmallDateTime و Uniqueidentifier و Time را نمی توان فشرده کرد.

نکته: اگر فضای مربوط به این عدد ۱۲ رو که از چهار بایت به یک بایت رسونده، شما بیایید عدد ۱۲ را بنا به هر دلیلی به عدد ۱۲۰۰ تغییر دهید در این صورت Page Split رخ داده و باعث کاهش Performance می شود. به خاطر همین است که توصیه بر این است که سعی کنید روی دیتاهایی که Report ایی و یا آرشیو هستند عملیات فشرده سازی را انجام دهید.

Page Level Compression

در الگوریتم فشرده سازی به روش Page Level Compression، به طور پیش فرض Row Level Compression اتفاق می افتد.

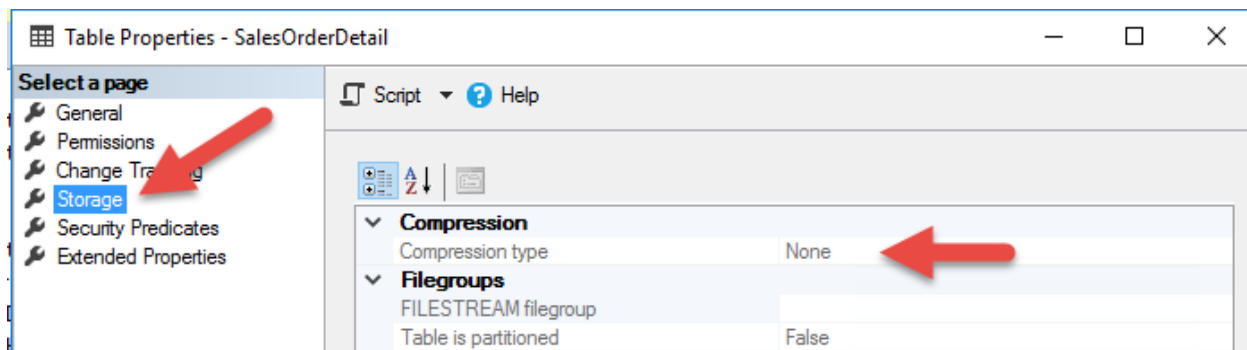
در ضمن دو تا Compression دیگر نیز در الگوریتم Page Level Compression اتفاق می افتد، یکی Prefix Column Compression و دیگری Dictionary Compression می باشد.

اگر در ستون LastName من ۲ تا فامیلی اکبری داشته باشم و فرض کنید که طراح دیتابیس دیتا تایپ این فیلد را به اشتباه Char(۳۰) گرفته باشد. قاعدتا باید ۶۰ بایت فضا اشغال می شد. ولی اگر شما عملیات فشرده سازی (به روش

Page) را انجام دهید در این صورت در قدم اول شما ۵۰ بایت شما فشرده سازی خواهید داشت، یعنی اول الگوریتم Row را اعمال می کند(به ازاء هر نام اکبری ، ۲۵ بایت فضا Save خواهید داشت). در قدم بعدی الگوریتم فشرده سازی به روش Page ، اسکیوال سرور می آد یک نام اکبری را نگه داشته و مابقی را بهش Reference می دهد(شبيه کاری که در Column Store Index انجام می شود). نتیجه اینکه در جداولی که کاردیتالیته دیتا در ستون ها پایین است (میزان تکرار داده ها بالاست (مثل ستون نام)) در این صورت نتیجه فشرده سازی بسیار متفاوت خواهد بود. تذکر : فشرده سازی روی ستون های BLOB چندان تاثیرگذار نیست.

اسکرپیت مربوط به فشرده سازی به روش Page Level Compressing

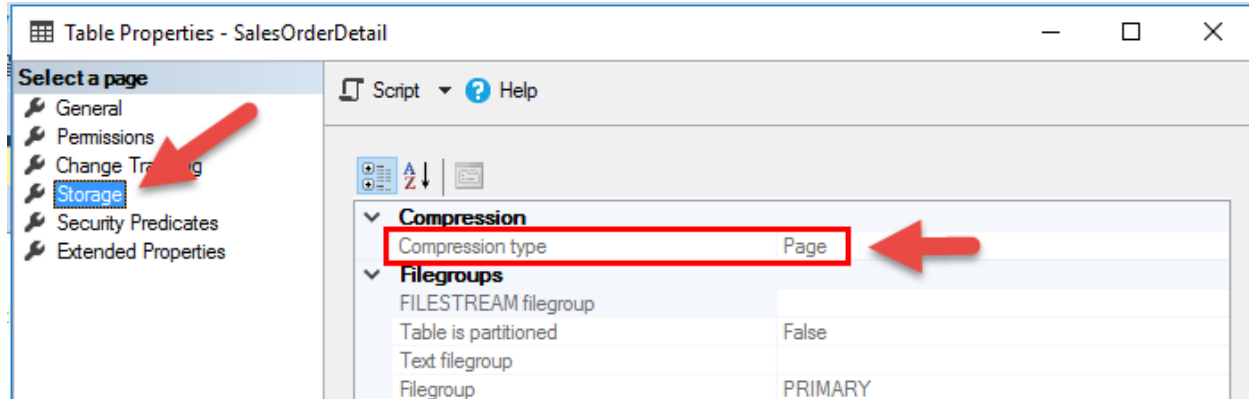
اگر بر روی جدول مورد نظرتان کلیک راست نموده و گزینه Properties را کلیک کنید و در پنجره ظاهر شده مطابق شکل زیر، بر روی Storage کلیک کنید در این صورت در قسمت Compression Type ، نوع فشرده سازی را خواهید دید، که در شکل زیر None می باشد.



حال چنانچه از اسکرپیت زیر برای فشرده سازی به روش Page Level Compressing استفاده کنید(در کد زیر در ضمن می خواهیم دو تا Core های سرورمان درگیر شوند)

```
ALTER TABLE [Sales].[SalesOrderDetail] Rebuild
WITH(DATA_COMPRESSION = PAGE,MAXDOP=۲)
```

در این صورت مطابق شکل زیر در قسمت Compression Type گزینه Page را خواهید دید. بعد از اجرای کد مربوط به فشرده سازی با الگوریتم Page Level Compression مطابق شکل زیر در قسمت Compression Type گزینه Page را خواهید دید.



اگر بنا به هر دلیلی از فشرده سازی جدول مورد نظرتان پشیمان شدید کفایت کد زیر را اجرا کنید :

```
ALTER TABLE [Sales].[SalesOrderDetail] Rebuild
WITH(DATA_COMPRESSION = NONE,MAXDOP=۲)
```

تذکر: شما می توانید ایندکس های نان کلاستر را نیز فشرده کنید. حتی شما می توانید فقط داده های مربوط به پارتیشن های خاصی را فشرده کنیم.

تذکر: اگر خواستید یک نان کلاستر ایندکس را فشرده کنید در این صورت می توانید مطابق کد زیر عمل نمایید. به مثال زیر دقت کنید.

```
ALTER INDEX IX_tblSales_OrderDate
ON tblSales REBUILD
WITH (DATA_COMPRESSION = PAGE)
```

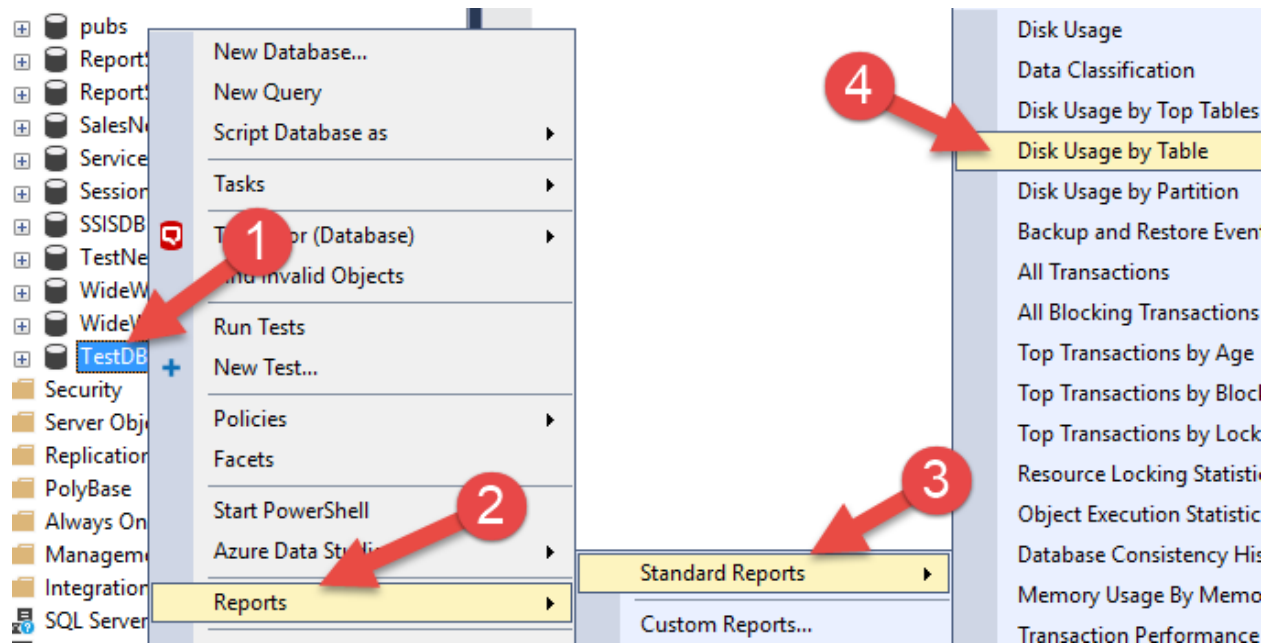
نحوه بدست آوردن آبجکت های فشرده شده:

برای بدست آوردن آبجکت های فشرده شده در یک دیتابیس می توانیم از کوئری زیر استفاده کنیم:

```
SELECT DISTINCT
    s.name,t.name,i.name,i.type,i.index_id,
    p.partition_number,p.rows
FROM sys.tables t
LEFT JOIN sys.indexes i
ON t.object_id = i.object_id
JOIN sys.schemas s
ON t.schema_id = s.schema_id
LEFT JOIN sys.partitions p
ON i.index_id = p.index_id
AND t.object_id = p.object_id
```

تذکر: در جداول دیتابیس های Data Warehouse هیچگاه جداول Fact Less Fact را فشرده نمی کنیم. این جدول که به آن Junction Table نیز گفته می شود حاصل ارتباط چند به چند بین جدول Fact و یک جدول Dimension است.
مثال: یک کپی از جدول tblSales با داده هایش با نام tblSales۲ بسازید. سپس جدول اول یعنی tblSales را با الگوریتم Page فشرده کنید.

مشاهده حجم اشغالی توسط دو جدول tblSales و tblSales۲ توسط گزارش های خود SQL Server



در این صورت پنجره ایی مطابق شکل زیر نمایان می شود.

Disk Usage by Table
[TestDB] SQL Server

on SQLSoftware at 4/20/2022 10:02:10 AM

This report provides detailed data on the utilization of disk space by tables within the Database. The report does not provide data for memory optimized tables.

Table Name	# Records	Reserved (KB)	Data (KB)	Indexes (KB)	Unused (KB)
dbo.tblSales	121,317	6,672	4,608	1,608	456
dbo.tblSales2	121,317	18,952	18,856	8	88

در قدم بعدی می خواهیم Cost دو کوئری را به کمک Execution Plan دو کوئری زیر را با هم مقایسه کنیم :
حال Actual Execution Plan رو فعال کنید و یک Select از دو جدول بگیرید در این صورت خواهید دید که Cost حالت فشرده ۲۳ درصد و Cost جدول غیرفشرده ۷۷ درصد می باشد.

```

SELECT SalesOrderID,SalesOrderDetailID,OrderQty
FROM dbo.tblSales
SELECT SalesOrderID,SalesOrderDetailID,OrderQty
FROM dbo.tblSales2
    
```

157 %

Results Messages Execution plan

Query 1: Query cost (relative to the batch): 23%

SELECT SalesOrderID,SalesOrderDetailID,OrderQty FROM dbo.tblSales

Table Scan [tblSales]
Cost: 100 %
0.200s
121317 of 121317 (100%)

Query 2: Query cost (relative to the batch): 77%

SELECT SalesOrderID,SalesOrderDetailID,OrderQty FROM dbo.tblSales2

Table Scan [tblSales2]
Cost: 100 %
0.117s
121317 of 121317 (100%)

در قدم بعدی می خواهیم مطابق شکل زیر به کمک دستور Set Statistics IO ON آمار مربوط به تعداد Physical Read و Logical Read مربوط به دو جدول را بدست آورده و با هم مقایسه کنیم. همانطور که مشاهده می کنید آمار تعداد IO های مربوط به Logical Reads مربوط به جدول فشرده ۵۷۶ عدد می باشد (یعنی عدد Page از هارد دیسک خوانده و آورده به حافظه)، و آمار تعداد IO های مربوط به Logical Reads مربوط به جدول عادی ۲۳۵۷ عدد می باشد.

```

SQLQuery32.sql - 1...B (OCS0\00623 (97))* - X SQLQuery30.sql - 1...B (OCS0\00623 (58))* SQLQuery29.sql - 1...B (OCS0\00623 (57))*
SET STATISTICS IO ON
SELECT SalesOrderID,SalesOrderDetailID,OrderQty
FROM dbo.tblSales
SELECT SalesOrderID,SalesOrderDetailID,OrderQty
FROM dbo.tblSales2
SET STATISTICS IO OFF
    
```

157 %

Results Messages

(121317 rows affected)
Table 'tblSales'. Scan count 1, logical reads 576, physical reads 0, logical reads in memory 0, physical reads in memory 0

(121317 rows affected)
Table 'tblSales2'. Scan count 1, logical reads 2357, physical reads 0, logical reads in memory 0, physical reads in memory 0

Completion time: 2022-04-20T10:38:01.9590502+04:30

ولی به یک نکته توجه کنید و آن اینکه Cost ، CPU Usage مربوط به جدول فشرده مطابق شکل زیر بالاتر می رود. که این کاملا طبیعی است.

```

SQLQuery30.sql - 1...B (OCS0\00623 (58))*
SQLQuery29.sql - 1...B (OCS0\00623 (69))*
SQLQuery28

SET STATISTICS TIME ON
SELECT SalesOrderID, SalesOrderDetailID, OrderQty
FROM dbo.tblSales
SELECT SalesOrderID, SalesOrderDetailID, OrderQty
FROM dbo.tblSales2
SET STATISTICS TIME OFF

157 %
Results Messages

(121317 rows affected)

SQL Server Execution Times:
    CPU time = 188 ms, elapsed time = 717 ms.

(121317 rows affected)

SQL Server Execution Times:
    CPU time = 125 ms, elapsed time = 2353 ms.

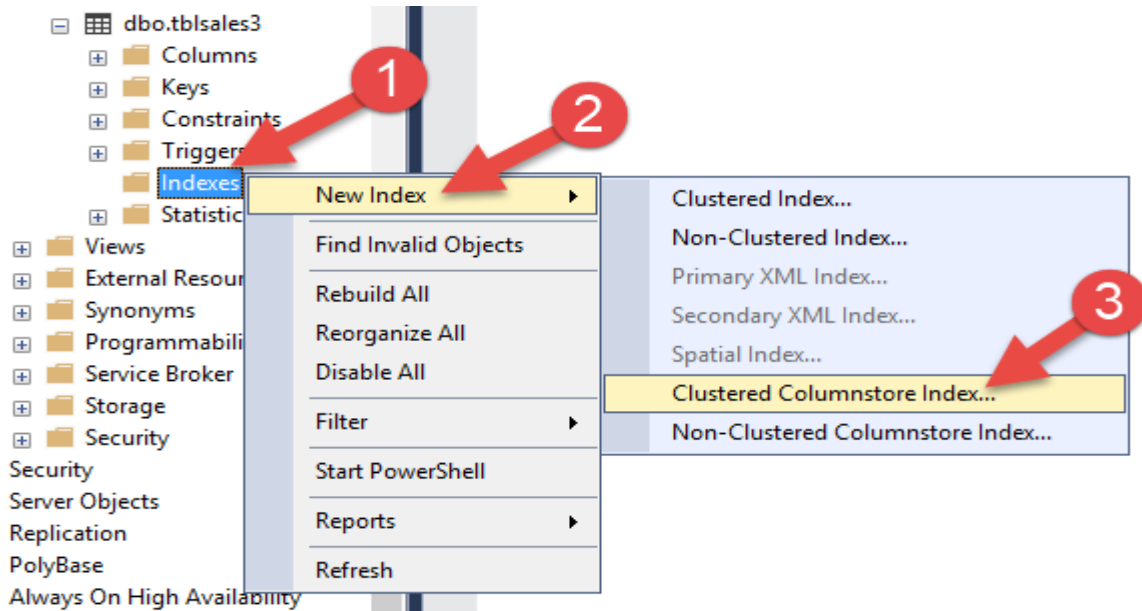
Completion time: 2022-04-20T09:18:30.8386041+04:30
    
```

در مرحله بعدی این مثال یک جدول سومی درست کنید و داده های آن را مطابق کوئری زیر پر کنید و سپس روی آن یک Column Store Index درست کنید.

```

SELECT *
INTO tblsales3
FROM tblsales2
    
```

کافی در جدول جدید یعنی tblSales^۳ که یک جدول Heap است کلیک راست کرده و گزینه New Index و سپس گزینه Clustered Columnstore Index را مطابق شکل زیر کلیک می نمایم.



حال اگر بخواهیم حجم سه جدول را با هم مقایسه کنیم ، شکلی مطابق شکل زیر خواهیم داشت.

SQLQuery37.sql - 1...(OCS0\00623 (102))* SQLQuery36.sql - 1...(OC

```
EXEC sp_spaceused 'tblSales'
EXEC sp_spaceused 'tblSales2'
EXEC sp_spaceused 'tblSales3'
```

157 %

	name	rows	reserved	data	index_size	unused
1	tblSales	121317	6672 KB	4608 KB	1608 KB	456 KB
1	tblSales2	121317	18952 KB	18856 KB	8 KB	88 KB
1	tblsales3	121317	18952 KB	18856 KB	8 KB	88 KB

حال اگر Cost سه جدول را با استفاده از Execution Plan مشاهده کنیم ، مطابق شکل زیر خواهیم دید که روی جدول سوم که Column Store Index زدیم ، **میزان Cost** ، **یک درصد** می باشد و جدول اول که با الگوریتم Page فشرده شده است میزان Cost آن ۲۳ درصد شده است و جدول عادی یعنی tblSales میزان Cost اش ۷۶ درصد می باشد.

The screenshot displays the SQL Server Enterprise Manager interface with three query windows open. The 'Results' tab is selected, showing the execution plan for each query. Red boxes and arrows highlight the relative query costs:

- Query 1:** Query cost (relative to the batch): 23%. Execution plan: Table Scan [tblSales1].
- Query 2:** Query cost (relative to the batch): 76%. Execution plan: Table Scan [tblSales2].
- Query 3:** Query cost (relative to the batch): 1%. Execution plan: Columnstore Index Scan (Clustered) [tblsales3].[ClusteredColumnStoreIn...].

نکته مهم: اگر شما مفهوم Compression را با مفهومی به نام Partition بندی ترکیب کنید که بسیار عالی خواهد شد.