В Management Studio планы выполнения запросов (предполагаемый и фактический) можно посмотреть в удобной графической форме, для этого нужно нажать соответствующую кнопку на панели инструментов при открытом окне запросов (рис. 1). При этом для предполагаемого плана не нужно запускать запрос на выполнение.

В плане с помощью древовидной структуры показан поток данных (в виде стрелок) и последовательность преобразующих его операторов (в виде значков). Толщина стрелок показывает объем обрабатываемых данных. Читать его нужно справа налево и сверху вниз. Если навести курсор на значок оператора, то появится список сведений о нем, также можно посмотреть подробности, щелкнув на значке правой кнопкой мыши и выбрав **Свойства** из контекстного меню.

Справку по всему списку доступных сведений об операторах можно посмотреть в электронной документации по ссылке: <https://technet.microsoft.com/ru-ru/library/ms178071(v=sql.105).aspx>. Список всех операторов: <https://msdn.microsoft.com/ru-ru/library/ms175913(v=sql.105).aspx>.

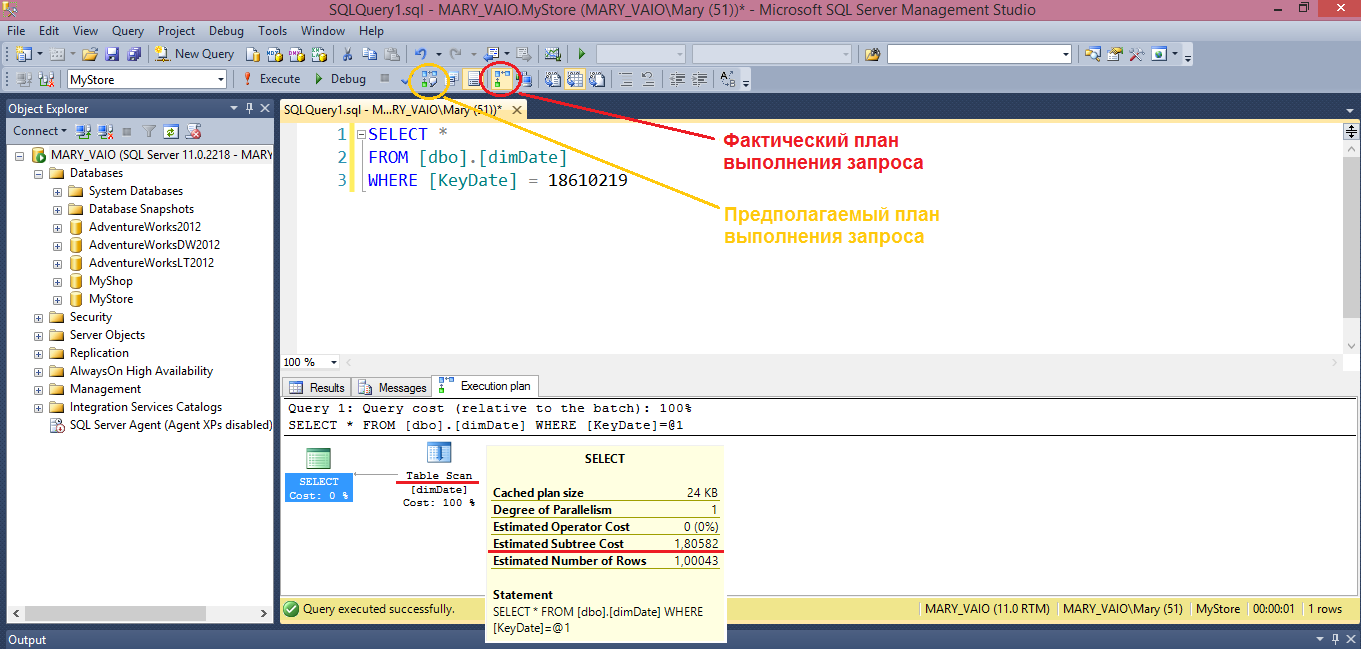


Рис. 1. План выполнения запроса к таблице, не имеющей индексов

Под каждой пиктограммой оператора показана его стоимость в процентах от общей стоимости запроса. Это число помогает понять, какая операция использует больше всего ресурсов.

Оценить примерные затраты на выполнение данной операции и всех предшествующих операций в поддереве можно с помощью показателя **Предполагаемая стоимость поддерева** (Estimated Subtree Cost).

Для определения примерной общей стоимости всего запроса нужно выбрать корневую операцию (слева в графическом плане), будет показана суммарная стоимость всех операций.

Из рис. 1 видно, что производится сканирование всей таблицы (операция scan) и стоимость запроса ESC = 1,80582.

Эта стоимость является весьма относительным показателем производительности. Она измеряется в условных единицах и показывает предполагаемое время выполнения запроса на некой «эталонной аппаратной конфигурации». Опираясь на стоимость, оптимизатор запросов старается построить такой план, который потребует от сервера наименьшего числа ресурсов. Реальные аппаратные конфигурации могут значительно отличаться, кроме того, следует учитывать возможную неактуальность статистики или ограничения модели оценки и т.д. Поэтому при оценке производительности запросов на реальной базе данных следует пользоваться более информативными метриками, предоставляемыми SQL Server.

Основные показатели, на которых следует сфокусировать внимание – это количество процессорного времени (CPU) для обработки кода запроса и операций чтения/записи (IO) из КЭШа данных (логических чтений) и, если данных нет в КЭШе, то физических чтений с диска. Предполагаемую стоимость следует учитывать для предварительной оценки запросов.

Таким образом, задача увеличения производительности сводится к задаче уменьшения операций чтения с диска и использования ресурсов центрального процессора.

Для замера реального времени выполнения запроса можно применить следующие подходы.

* Используя функции SQL Server работы со временем, вычислять разницу между системным временем до и после выполнения запроса.
* Использовать инструкции Transact-SQL для исследования статистики, включив их перед выполнением запроса. Установки повлияют только на текущий сеанс и позволят SQL Server выводить информацию о производительности на вкладку **Сообщения** (Messages).

SET STATISTICS IO ON – отображает сведения о взаимодействии SQL Server с диском во время выполнения запроса. Наиболее полезными сведениями здесь являются: число страниц, считанных из КЭШа данных, число страниц, считанных с диска и число страниц, помещенных в кэш для запроса (упреждающее чтение).

SET STATISTICS TIME ON – отображает процессорное время в миллисекундах, которое было использовано для синтаксического анализа, компиляции и выполнения каждой инструкции. Этот хороший способ измерения, так как это серверная метрика. На рис. 2 показан пример использования этих инструкций.

* Использовать специальную утилиту SQL Profiler, которая представляет собой развитый интерфейс, предназначенный для создания, анализа и управления трассировками. Все события сохраняются в файле трассировки, который затем может быть проанализирован.

Для запуска приложения из меню среды SQL Server Management Studio выберите **Инструменты** (Tools) -> SQL Server Profiler или в редакторе запросов щелкните правой кнопкой мыши, а затем выберите пункт **Трассировка запроса в приложении SQL Server Profiler** (Trace Query in SQL Server Profiler).

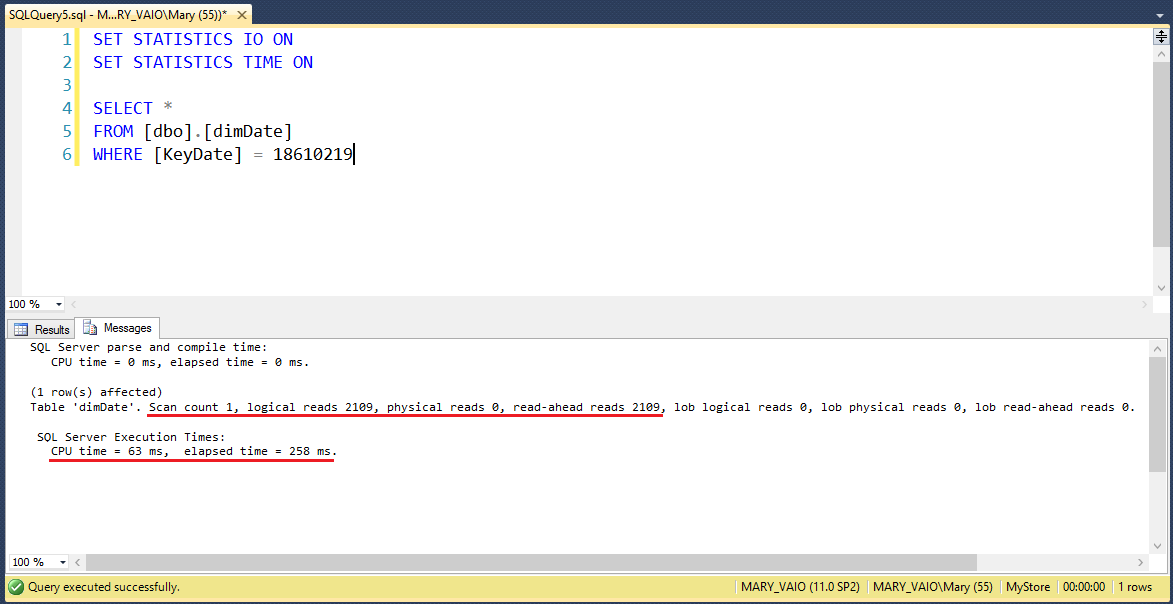


Рис. 2. Исследование статистики запроса с помощью инструкций Transact-SQL

Однако следует помнить, что все эти способы измерения времени выполнения запроса сильно зависят от окружающей среды проведения теста.

Проверить занимаемое таблицей место на диске можно с помощью стандартного отчета **Использование диска таблицей** (Disk Usage by Table) (рис. 3). Для его создания щелкните правой кнопкой мыши на названии базы данных и выберите из контекстного меню **Reports -> Standard Reports -> Disk Usage by Table.**

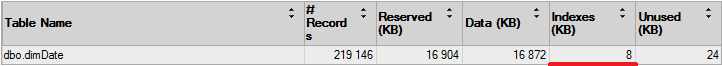
****

Рис. 3. Количество дискового пространства, занимаемого таблицей «dimDate»

Из рисунка видно, что под индекс стандартно выделена одна страница данных, но сам индекс не занимает место на диске.

Теперь создадим кластеризованный индекс и посмотрим, как он изменит стоимость предыдущего запроса и занимаемое таблицей место на диске. Для этого выполним инструкцию T-SQL и проверим, как изменился объем занимаемого таблицей места на диске (рис. 4).

CREATE CLUSTERED INDEX CL\_IndexDate

ON [dbo].[dimDate]([KeyDate])

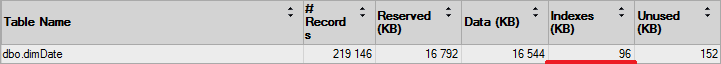


Рис. 4. Количество дискового пространства, занимаемого таблицей «dimDate»

после создания кластеризованного индекса

Из рисунка видно, что созданный индекс не занимает много места на диске. Выполним тот же запрос, но уже на индексированной таблице и посмотрим, как изменилась его стоимость (рис. 5). Так как условие отбора задано на столбце, являющимся ключом кластеризованного индекса, то время поиска строки заметно сократилось, ESC = 0,0032831. Операция **scan** сменилась на операцию **seek**. Теперь не нужно просматривать все строки таблицы, а достаточно спуститься по нужной ветке дерева до соответствующего листа, где хранятся данные.

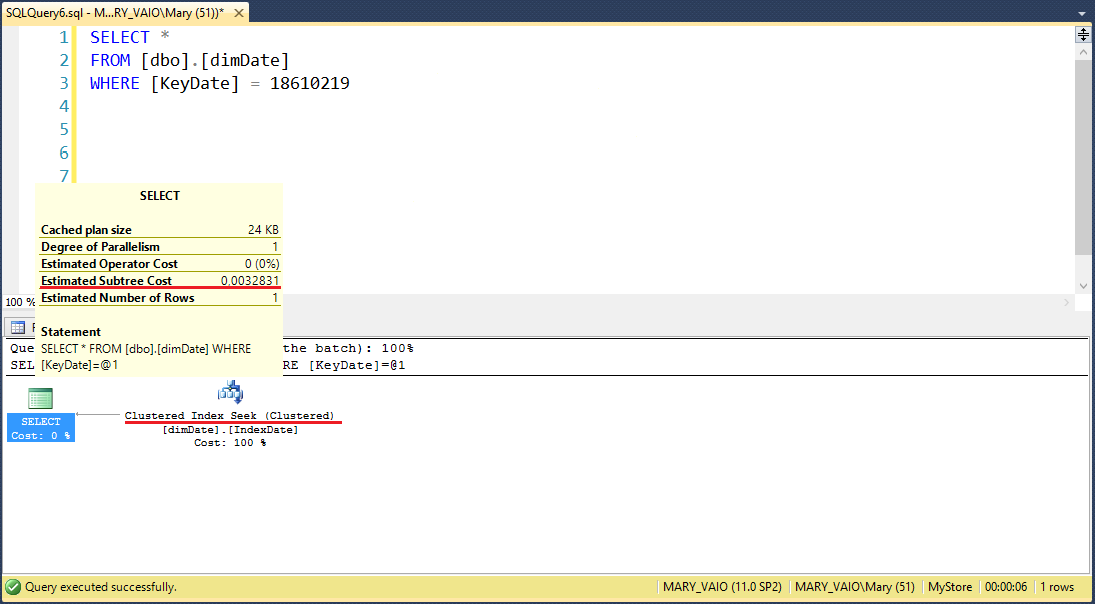


Рис. 38. Поиск данных по ключу кластеризованного индекса таблицы «dimDate»

Теперь выполним поиск строк не по ключевому полю индекса, например:

SELECT \*

FROM [dbo].[dimDate]

WHERE [Year]=2015

В результате увидим, что сервер снова выполняет сканирование всей таблицы (рис. 6) и стоимость запроса вновь возрастет.

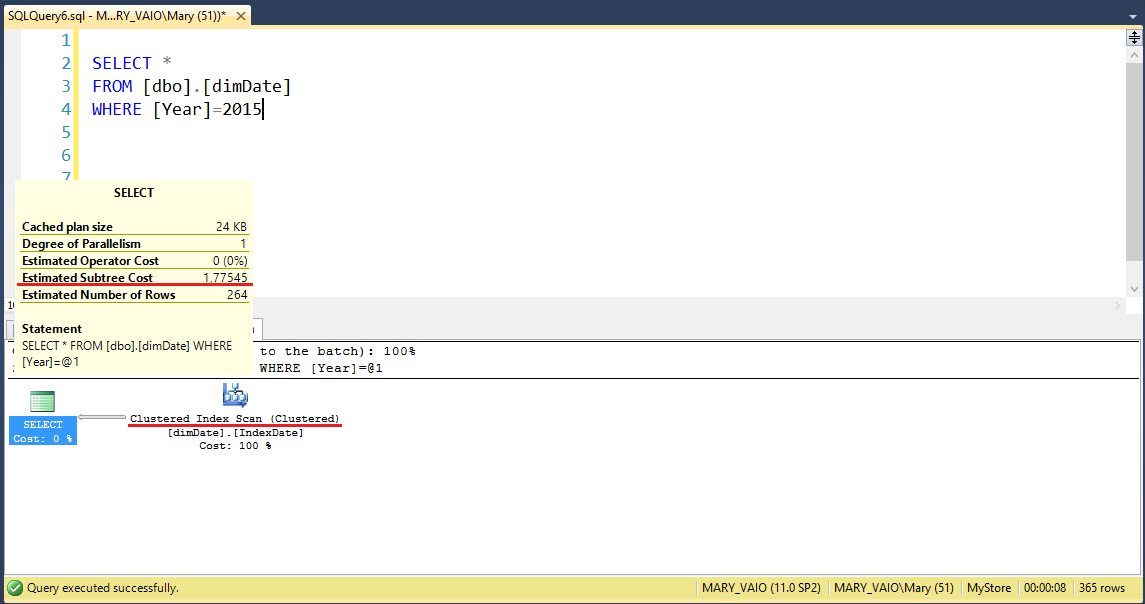


Рис. 6. Поиск данных по полю, не входящему в ключевые поля

кластеризованного индекса таблицы «dimDate»

Для ускорения поиска по полям, не входящим в кластеризованный индекс, можно применить механизм некластеризованных индексов.

Создадим некластеризованный индекс по полю «Year» для таблицы «dimDate» и посмотрим, сколько теперь будет занимать места эта таблица на диске (рис. 7).

CREATE INDEX NonCL\_Index\_Year

on [dbo].[dimDate]([Year])

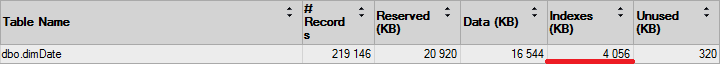


Рис. 7. Количество дискового пространства, занимаемого таблицей «dimDate»

после создания некластеризованного индекса

Как видно из рисунка, данный индекс занимает уже ощутимое место на диске – 4056Кб, это почти 25% от всего объема занимаемого таблицей.

Некластеризованные индексы повышают производительность запросов, возвращающих значения из ключевых полей этого индекса.

Например, такой запрос:

SELECT [KeyDate], [Year]

FROM [dbo].[dimDate]

WHERE [Year]=2015

будет иметь стоимость ESC = 0,0036835, но какова будет стоимость запроса, если необходимо выбрать данные, не входящие в некластеризованный индекс.

Снова выполним запрос:

SELECT \*

FROM [dbo].[dimDate]

WHERE [Year]=2015

и посмотрим на его стоимость и план выполнения (рис. 8).

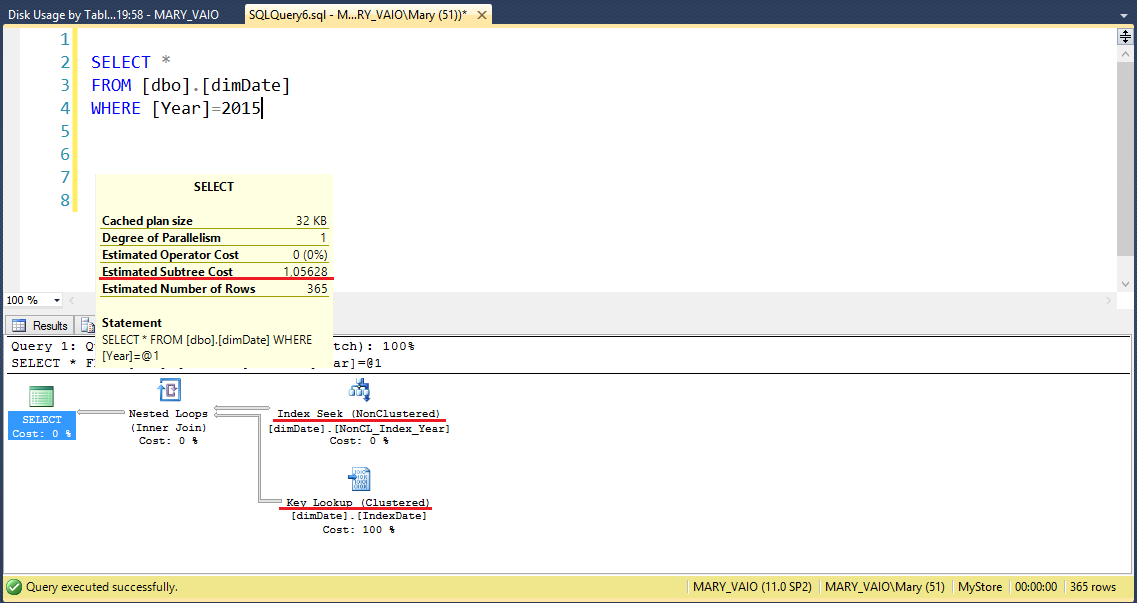


Рис. 8. Поиск по низкоселективному столбцу «Year»

некластеризованного индекса таблицы «dimDate»

Из рисунка видно, что в данном случае преимущество использования некластеризованного индекса невелико, ESC = 1,05628. Стоимость запроса ненамного уменьшилась при использовании некластеризованного индекса, а он, в свою очередь, занимает много места. Связано это с тем, что столбец, на котором задан некластеризованный индекс, содержит множество повторяющихся значений (*низкоселективных данных*) и требует многократного поиска данных по кластеризованному индексу. Отсюда вывод: некластеризованные индексы, заданные на низкоселективных столбцах часто не эффективны и стоимость запросов с их использованием может превысить стоимость простого сканирования таблицы. В таком случае сервер будет выполнять именно его.

Например, если задать некластеризованный индекс для низкоселективного столбца «Week» таблицы «dimDate», и выполнить поиск по нему, с выводом столбцов, не входящих в индекс, то сервер будет выполнять сканирование таблицы, игнорируя этот индекс (рис. 9).

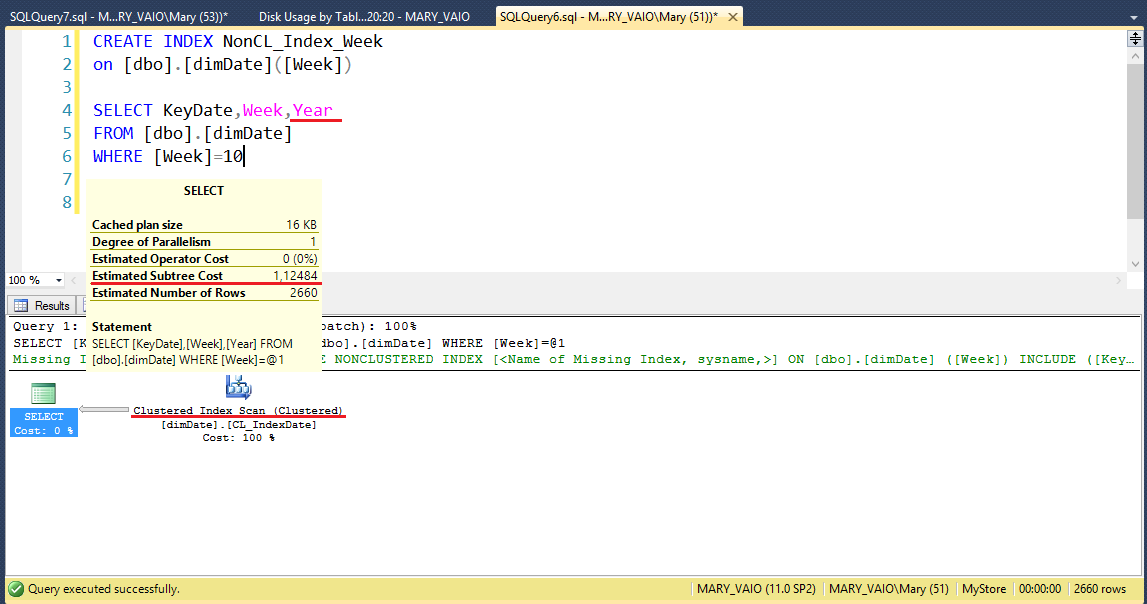


Рис. 9. Операция сканирования таблицы без использования индекса

при поиске по низкоселективному столбцу «Week»

Алгоритм принятия решения основан на статистике о селективности столбцов, которую ведет сервер для каждой таблицы. Объект статистики для таблицы создается по индексу или списку столбцов таблицы. Для просмотра свойств этого объекта из среды SQL Server Management Studio в **Обозревателе объектов** нужно выбрать таблицу и в её раскрывающемся списке папок найти папку **Статистика** (Statistics). Затем, щелкнув правой кнопкой мыши на объекте статистики, выбрать команду **Свойства** (Properties). Свойства включают заголовок, содержащий метаданные о статистике, гистограмму, содержащую распределение значений в первом ключевом столбце объекта статистики, и вектор плотностей для измерения корреляции с охватом нескольких столбцов.  На рис. 10 показаны свойства для статистики по некластеризованному индексу, заданному по полю «Year».

Следующие данные описывают столбцы, возвращенные в результирующем наборе для гистограммы.

RANGE\_HI\_KEY – верхнее граничное значение столбца для шага гистограммы. Это значение столбца называется также ключевым значением.

RANGE\_ROWS – предполагаемое количество строк, значение столбцов которых находится в пределах шага гистограммы, исключая верхнюю границу.

EQ\_ROWS – предполагаемое количество строк, значение столбцов которых равно верхней границе шага гистограммы.

DISTINCT\_RANGE\_ROWS – предполагаемое количество строк с различающимся значением столбца в пределах шага гистограммы, исключая верхнюю границу.

AVG\_RANGE\_ROWS – среднее количество строк с повторяющимися значениями столбцов в пределах шага гистограммы, исключая верхнюю границу (RANGE\_ROWS/DISTINCT\_RANGE\_ROWS для DISTINCT\_RANGE\_ROWS > 0).

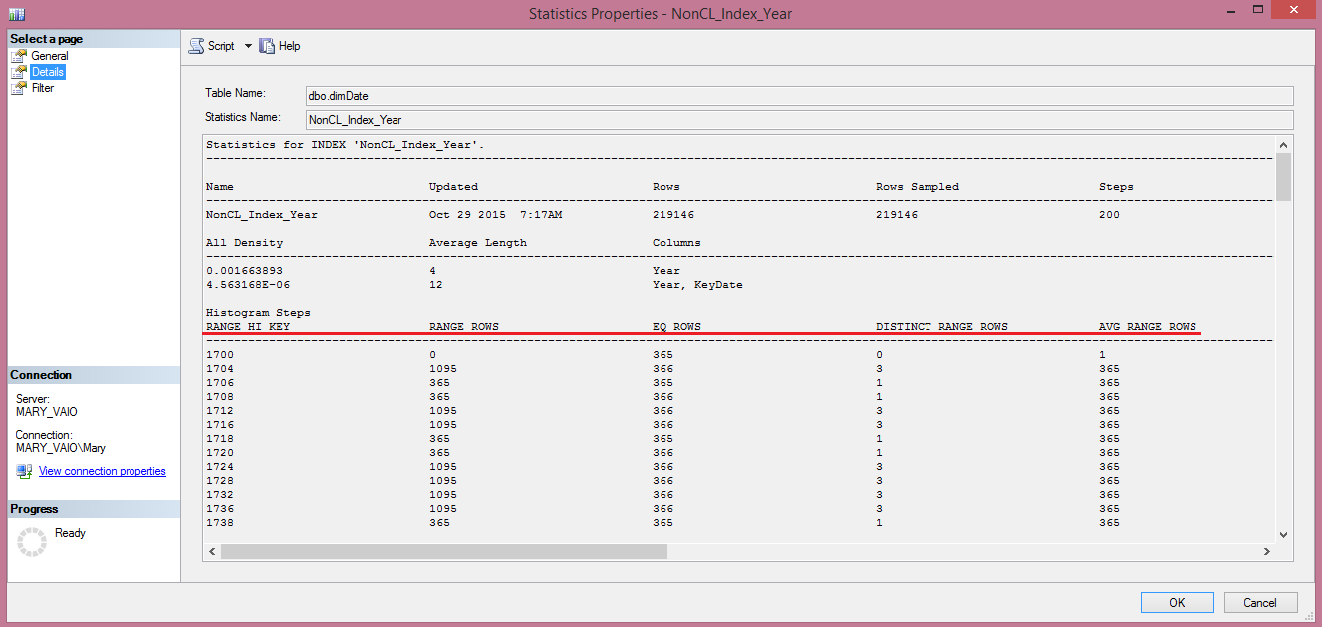


Рис. 10. Свойства статистики для некластеризованного индекса по полю «Year»

Более подробно о свойствах статистики можно узнать в электронной документации по ссылке <https://technet.microsoft.com/ru-ru/library/hh510179(v=sql.110).aspx>.

Перед тем, как построить некластеризованный индекс, разработчик должен подумать о селективности столбцов, на которых он будет построен.

Если индексировать высокоселективные столбцы (мало повторяющихся значений), то выигрыш по стоимости запросов, содержащих условие отбора по этому столбцу, очевиден.