1 Создать таблицу для тестирования запросов, содержащие большое количество записей, более 10000 записей.

Код запроса

|  |
| --- |
| declare  @idWorker int,  @idRoom int = -1,  @idClient int,  @idReservation int = null,  @priceForLiving money,  @priceForServices money = 0,  @fine money = 0,  @planingLeaveDate smalldatetime,  @arriveDate smalldatetime,  @leaveDate smalldatetime,  @countOfPeople tinyint,  @countOfChildren tinyint = 0;    declare @startDate smalldatetime = '01.01.2000 13:00:00';  declare  @min int,  @max int;  declare  @day int,  @hour int,  @minutes int;  declare  @start int = 0,  @end int = 10000;  select top 1 @startDate = Дата\_заезда from Заезд order by id\_Заезд desc  set @startDate = dateadd(month, 1, @startDate)  while @start < @end begin  -- номера  if (exists (  SELECT TOP 1 \* from Номер  where СостояниеНомера = N'готов к эксплуатации'  ORDER BY NEWID()  )) begin  SELECT TOP 1 @idRoom = idНомера from Номер  where СостояниеНомера = N'готов к эксплуатации'  ORDER BY NEWID()  update Номер set СостояниеНомера = N'эксплуатируется' where idНомера = @idRoom  -- сотрудники  SELECT TOP 1 @idWorker = id\_Сотрудники from Сотрудники  where Должность = N'Регистратор' ORDER BY NEWID()  -- клиенты  SELECT TOP 1 @idClient = id\_Клиент from Клиент ORDER BY NEWID()  -- Дата\_заезда  set @min = 1;  set @max = 28;  set @day = round((rand() \* @max) + @min, 0);  set @min = 1;  set @max = 10 - @min + 1;  set @hour = round((rand() \* @max) + @min, 0);  set @arriveDate = @startDate;  set @arriveDate = dateadd(DAY, @day, @arriveDate)  set @arriveDate = dateadd(hour, @hour, @arriveDate)  -- Запланированная\_дата\_выезда  set @planingLeaveDate = @arriveDate;  set @min = 1;  set @max = 28;  set @day = round((rand() \* @max) + @min, 0);  set @planingLeaveDate = dateadd(DAY, @day, @arriveDate)  set @planingLeaveDate = dateadd(hour, -@hour, @planingLeaveDate)  set @min = 1;  set @max = 10 - @min + 1;  set @hour = round((rand() \* @max) + @min, 0);  set @planingLeaveDate = dateadd(hour, @hour, @planingLeaveDate)  -- Дата\_выезда  set @LeaveDate = @planingLeaveDate;  -- Количество\_человек  set @min = 1;  set @max = 3 - @min;  set @countOfPeople = round((rand() \* @max) + @min, 0);  -- цена за проживание  set @priceForLiving = dateDiff(DAY, @arriveDate, @leaveDate) + 1 -- количество прожитых дней  -- расчет цены за проживание  select @priceForLiving \*= Цена from Номер  where idНомера = @idRoom  INSERT INTO [dbo].[Заезд]  ([Сотрудники\_idСотрудники]  ,[Номер\_idНомера]  ,[Клиент\_id\_Клиент]  ,[Бронирование\_idБронирование]  ,[Цена\_за\_проживание]  ,[Цена\_за\_дополнительные\_услуги]  ,[Размер\_штрафа]  ,[Запланированная\_дата\_выезда]  ,[Дата\_заезда]  ,[Дата\_выезда]  ,[Количество\_человек]  ,[Количество\_детей])  VALUES  (@idWorker  ,@idRoom  ,@idClient  ,@idReservation  ,@priceForLiving  ,@priceForServices  ,@fine  ,@planingLeaveDate  ,@arriveDate  ,@LeaveDate  ,@countOfPeople  ,@countOfChildren)  end else begin  set @startDate = dateadd(month, 1, @startDate)  update Номер set СостояниеНомера = N'готов к эксплуатации'  continue  end  set @start += 1;  end  INSERT INTO [dbo].[Статистика]  ([ФИО\_Клиента]  ,[НомерКомнаты]  ,[ДлительностьПроживания]  ,[ДатаЗаезда]  ,[ДатаВыезда]  ,[СуммаЗаПроживание]  ,idЗаезд)  select  ФИО\_клиента,  НомерКомнаты,  dateDiff(DAY, Дата\_заезда, Дата\_выезда) + 1,  Дата\_заезда,  Дата\_выезда,  Цена\_за\_проживание+Цена\_за\_дополнительные\_услуги,  id\_Заезд  from Заезд  join Клиент on Клиент\_id\_Клиент=id\_Клиент  join Номер on idНомера=Номер\_idНомера |

2 Инструкции Set

Пример 19.3 Отображение текстового плана запроса посредством опции SHOWPLAN\_TEXT

Код запроса

|  |
| --- |
| SET SHOWPLAN\_TEXT ON;  GO  SELECT \* FROM Заезд  JOIN Статистика ON Заезд.id\_Заезд = Статистика.idЗаезд and Заезд.id\_Заезд = 90000  GO  SET SHOWPLAN\_TEXT OFF; |

Результат запроса

|  |
| --- |
| |--Nested Loops(Inner Join)  |--Clustered Index Seek(OBJECT:([Hotel].[dbo].[Заезд].[PK\_\_Заезд\_\_D05F06D1EA809C2B]), SEEK:([Hotel].[dbo].[Заезд].[id\_Заезд]=(90000)) ORDERED FORWARD)  |--Clustered Index Scan(OBJECT:([Hotel].[dbo].[Статистика].[PK\_Статистика]), WHERE:([Hotel].[dbo].[Статистика].[idЗаезд]=(90000))) |

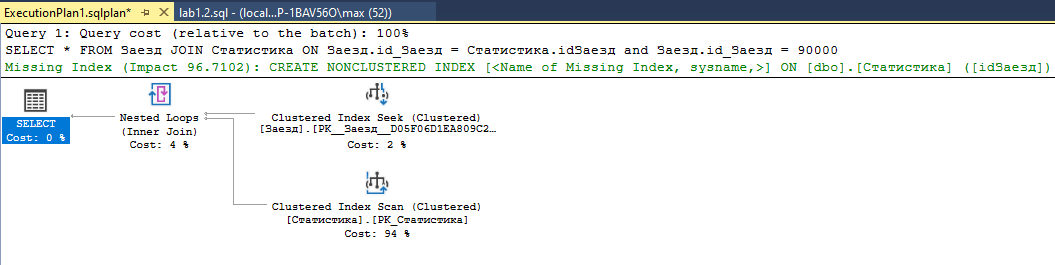
Пример 19.4 Отображение плана выполнения посредством опции SHOWPLAN\_XML

Код запроса

|  |
| --- |
| SET SHOWPLAN\_XML ON;  GO  SELECT \* FROM Заезд  JOIN Статистика ON Заезд.id\_Заезд = Статистика.idЗаезд and Заезд.id\_Заезд = 90000  GO  SET SHOWPLAN\_XML OFF;  GO |

Результат запроса



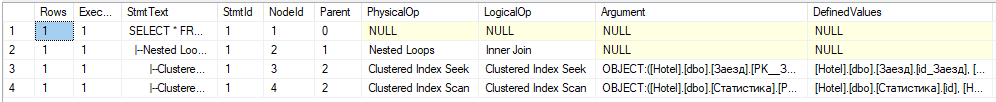


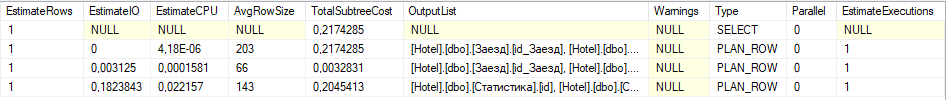
Пример 19.5 Использование опции STATISTICS PROFILE в инструкции SET

Код запроса

|  |
| --- |
| SET STATISTICS PROFILE ON;  GO  SELECT \* FROM Заезд  JOIN Статистика ON Заезд.id\_Заезд = Статистика.idЗаезд and Заезд.id\_Заезд = 90000  GO  SET STATISTICS PROFILE OFF; |

Результат запроса





Задание 3

Для вывода времени сообщением была использована команда set statistics time on. Также была использована команда set showplan\_text on для вывода текстового плана запроса.

6 запросов с выражением в условии отбора:

3.1. не правильное - содержит <>

Код запроса

|  |
| --- |
| SELECT \*  FROM [dbo].[Статистика]  WHERE ДлительностьПроживания <> 15 |

Текстовый план запроса

|  |
| --- |
| |--Clustered Index Scan(OBJECT:([Hotel].[dbo].[Статистика].[PK\_Статистика]), WHERE:([Hotel].[dbo].[Статистика].[ДлительностьПроживания]<>(15))) |

Время выполнения запроса

|  |
| --- |
| SQL Server Execution Times:  CPU time = 0 ms, elapsed time = 138 ms. |

3.2 правильное - заменить сравнение на >= и/или <

Код запроса

|  |
| --- |
| SELECT \*  FROM Статистика  WHERE ДлительностьПроживания >= 15 and ДлительностьПроживания < 20 |

Текстовый план запроса

|  |
| --- |
| |--Clustered Index Scan(OBJECT:([Hotel].[dbo].[Статистика].[PK\_Статистика]), WHERE:([Hotel].[dbo].[Статистика].[ДлительностьПроживания]>=(15) AND [Hotel].[dbo].[Статистика].[ДлительностьПроживания]<(20))) |

Время выполнения запроса

|  |
| --- |
| SQL Server Execution Times:  CPU time = 0 ms, elapsed time = 110 ms. |

3.3 не правильное - выражение слева от сравнения константа справа

Код запроса

|  |
| --- |
| SELECT \*  FROM Статистика  WHERE СуммаЗаПроживание\*1.15>100000 |

Текстовый план запроса

|  |
| --- |
| |--Clustered Index Scan(OBJECT:([Hotel].[dbo].[Статистика].[PK\_Статистика]), WHERE:(CONVERT\_IMPLICIT(numeric(19,4),[Hotel].[dbo].[Статистика].[СуммаЗаПроживание],0)\*(1.15)>(100000.000000))) |

Время выполнения запроса

|  |
| --- |
| SQL Server Execution Times:  CPU time = 15 ms, elapsed time = 105 ms. |

3.4 правильное - преобразовать выражение так, чтобы слева было поле, справа выражение

Код запроса

|  |
| --- |
| SELECT \*  FROM Статистика  WHERE СуммаЗаПроживание>100000/1.15 |

Текстовый план запроса

|  |
| --- |
| |--Clustered Index Scan(OBJECT:([Hotel].[dbo].[Статистика].[PK\_Статистика]), WHERE:([Hotel].[dbo].[Статистика].[СуммаЗаПроживание]>CONVERT\_IMPLICIT(numeric(10,0),[@1],0)/[@2])) |

Время выполнения запроса

|  |
| --- |
| SQL Server Execution Times:  CPU time = 0 ms, elapsed time = 110 ms. |

3.5 не правильное - содержит NOT IN

Код запроса

|  |
| --- |
| SELECT \*  FROM Статистика  WHERE ФИО\_Клиента not in (N'Архипов Емельян Арсеньевич', N'Рыбаков Климент Юлианович', N'Лобанов Андрей Тимурович') |

Текстовый план запроса

|  |
| --- |
| |--Clustered Index Scan(OBJECT:([Hotel].[dbo].[Статистика].[PK\_Статистика]), WHERE:([Hotel].[dbo].[Статистика].[ФИО\_Клиента]<>N'Архипов Емельян Арсеньевич' AND [Hotel].[dbo].[Статистика].[ФИО\_Клиента]<>N'Лобанов Андрей Тимурович' AND [Hotel].[dbo].[Статистика].[ФИО\_Клиента]<>N'Рыбаков Климент Юлианович')) |

Время выполнения запроса

|  |
| --- |
| SQL Server Execution Times:  CPU time = 16 ms, elapsed time = 196 ms. |

3.6 правильное - преобразовать выражение, чтобы не содержало NOT

Код запроса

|  |
| --- |
| SELECT \*  FROM Статистика  WHERE ФИО\_Клиента in (N'Архипов Емельян Арсеньевич', N'Рыбаков Климент Юлианович', N'Лобанов Андрей Тимурович') |

Текстовый план запроса

|  |
| --- |
| |--Clustered Index Scan(OBJECT:([Hotel].[dbo].[Статистика].[PK\_Статистика]), WHERE:([Hotel].[dbo].[Статистика].[ФИО\_Клиента]=N'Архипов Емельян Арсеньевич' OR [Hotel].[dbo].[Статистика].[ФИО\_Клиента]=N'Лобанов Андрей Тимурович' OR [Hotel].[dbo].[Статистика].[ФИО\_Клиента]=N'Рыбаков Климент Юлианович')) |

Время выполнения запроса

|  |
| --- |
| SQL Server Execution Times:  CPU time = 16 ms, elapsed time = 47 ms. |

Вывод. Из плана запросов и времени выполнения видно, что оптимизатор может использовать в качестве аргументов поиска любые типы выражений.

4 Индексы и статистика

Для вывода времени выполнения запроса сообщением была использована команда set statistics time on. Также была использована команда set statistics profile on для вывода текстового плана и времени выполнения запроса.

4.1 задания 19.7 - 19.9

4.1.1 Запрос с высоким уровнем селективности

Код запроса

|  |
| --- |
| select \* from Статистика  where ДлительностьПроживания = 1 |

Текстовый план запроса

|  |
| --- |
| |--Clustered Index Scan(OBJECT:([Hotel].[dbo].[Статистика].[PK\_Статистика]), WHERE:([Hotel].[dbo].[Статистика].[ДлительностьПроживания]=CONVERT\_IMPLICIT(int,[@1],0))) |

Время выполнения запроса

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



Время выполнения запроса через план запроса

4.1.2 Запрос с низким уровнем селективности

Код запроса

|  |
| --- |
| select \* from Статистика  where ДлительностьПроживания = 15 |

Текстовый план запроса

|  |
| --- |
| |--Clustered Index Scan(OBJECT:([Hotel].[dbo].[Статистика].[PK\_Статистика]), WHERE:([Hotel].[dbo].[Статистика].[ДлительностьПроживания]=CONVERT\_IMPLICIT(int,[@1],0))) |

Время выполнения запроса

|  |
| --- |
| SQL Server Execution Times:  CPU time = 0 ms, elapsed time = 128 ms. |



Время выполнения запроса через план запроса

4.1.3 Использование кластеризованного индекса

Код запроса

|  |
| --- |
| select \* from Статистика  where Статистика.id = 20015 |

Текстовый план запроса

|  |
| --- |
| |--Clustered Index Seek(OBJECT:([Hotel].[dbo].[Статистика].[PK\_Статистика]), SEEK:([Hotel].[dbo].[Статистика].[id]=CONVERT\_IMPLICIT(int,[@1],0)) ORDERED FORWARD) |

Время выполнения запроса

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



Время выполнения запроса через план запроса

4.2 создать запрос с условием на поле, которое не является ключом, ни первичным, ни внешним

|  |
| --- |
| select СуммаЗаПроживание from Статистика  where СуммаЗаПроживание > 100000 |

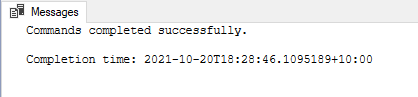
4.3 создать запрос с группировкой по столбцу, которое не является ключом, ни первичным, ни внешним

|  |
| --- |
| select ДлительностьПроживания from Статистика  where ДлительностьПроживания > 20  group by ДлительностьПроживания |

4.4 создать статистику по столбцу, используемому в условии и по столбцу, используемому в группировке

|  |
| --- |
| CREATE STATISTICS [S\_duration] ON [dbo].[МногоДанных]([ДлительностьПроживания])  GO  CREATE STATISTICS [S\_sum] ON [dbo].[МногоДанных]([СуммаЗаПроживание])  GO |

Результат выполнения



4.5 выполнить запрос 4.2 и 4.3

Текстовый план запроса

|  |
| --- |
| |--Clustered Index Scan(OBJECT:([Hotel].[dbo].[Статистика].[PK\_Статистика]), WHERE:([Hotel].[dbo].[Статистика].[СуммаЗаПроживание]>CONVERT\_IMPLICIT(money,[@1],0))) |

Время выполнения запроса

|  |
| --- |
| SQL Server Execution Times:  CPU time = 16 ms, elapsed time = 69 ms. |



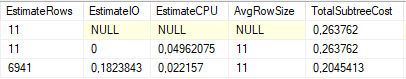
Время выполнения запроса через план запроса

Текстовый план запроса

|  |
| --- |
| |--Hash Match(Aggregate, HASH:([Hotel].[dbo].[Статистика].[ДлительностьПроживания]))  |--Clustered Index Scan(OBJECT:([Hotel].[dbo].[Статистика].[PK\_Статистика]), WHERE:([Hotel].[dbo].[Статистика].[ДлительностьПроживания]>(20))) |

Время выполнения запроса

|  |
| --- |
| SQL Server Execution Times:  CPU time = 0 ms, elapsed time = 2 ms. |

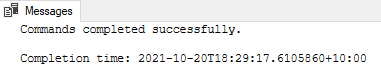


Время выполнения запроса через план запроса

4.6 очистить статистику.

|  |
| --- |
| DROP STATISTICS [dbo].[МногоДанных].[S\_duration]  GO  DROP STATISTICS [dbo].[МногоДанных].[S\_sum]  GO |

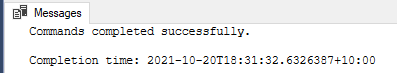
Результат выполнения



4.7 создать индекс по столбцам, используемым в условии и в группировке

|  |
| --- |
| CREATE INDEX IX\_Duration ON МногоДанных ([ДлительностьПроживания]);  CREATE INDEX IX\_Sum ON МногоДанных ([СуммаЗаПроживание]); |

Результат выполнения



4.8 выполнить запросы 4.2 и 4.3

Текстовый план запроса

|  |
| --- |
| |--Index Seek(OBJECT:([Hotel].[dbo].[Статистика].[IX\_Sum]), SEEK:([Hotel].[dbo].[Статистика].[СуммаЗаПроживание] > CONVERT\_IMPLICIT(money,[@1],0)) ORDERED FORWARD) |

Время выполнения запроса

|  |
| --- |
| SQL Server Execution Times:  CPU time = 0 ms, elapsed time = 59 ms. |



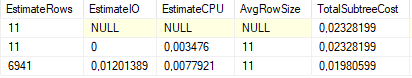
Время выполнения запроса через план запроса

Текстовый план запроса

|  |
| --- |
| |--Stream Aggregate(GROUP BY:([Hotel].[dbo].[Статистика].[ДлительностьПроживания]))  |--Index Seek(OBJECT:([Hotel].[dbo].[Статистика].[IX\_Duration]), SEEK:([Hotel].[dbo].[Статистика].[ДлительностьПроживания] > (20)) ORDERED FORWARD |

Время выполнения запроса

|  |
| --- |
| SQL Server Execution Times:  CPU time = 0 ms, elapsed time = 1 ms. |



Время выполнения запроса через план запроса

4.9 создать статистику по индексу на запросы из примеров 4.1.1 и 4.1.2

CREATE INDEX IX\_Duration ON Статистика ([ДлительностьПроживания]);

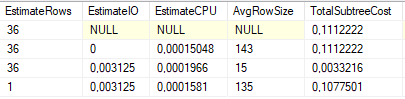
4.10 выполнить примеры 4.1.1 и 4.1.2

Текстовый план запроса

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

Время выполнения запроса

|  |
| --- |
| |--Nested Loops(Inner Join, OUTER REFERENCES:([Hotel].[dbo].[Статистика].[id], [Expr1003]) WITH UNORDERED PREFETCH)  |--Index Seek(OBJECT:([Hotel].[dbo].[Статистика].[IX\_Duration]), SEEK:([Hotel].[dbo].[Статистика].[ДлительностьПроживания]=(1)) ORDERED FORWARD)  |--Clustered Index Seek(OBJECT:([Hotel].[dbo].[Статистика].[PK\_Статистика]), SEEK:([Hotel].[dbo].[Статистика].[id]=[Hotel].[dbo].[Статистика].[id]) LOOKUP ORDERED FORWARD) |



Время выполнения запроса через план запроса

Текстовый план запроса

|  |
| --- |
| |--Clustered Index Scan(OBJECT:([Hotel].[dbo].[Статистика].[PK\_Статистика]), WHERE:([Hotel].[dbo].[Статистика].[ДлительностьПроживания]=(15))) |

Время выполнения запроса

|  |
| --- |
| SQL Server Execution Times:  CPU time = 0 ms, elapsed time = 111 ms. |



Время выполнения запроса через план запроса

Вывод. Исходя из результатов запросов с низкой и высокой селективностью видно, что время выполнения для высокой селективности занимает меньше времени, в то время как план выполнения схожий для обоих запросов.

Из времени выполнения между запросом с условием на поле и запросом с условием по столбцу, используемому в группировке, видно, что запрос с условием только на поле выполняется медленнее. Применение статистики без индексов и с индексами к этим запросам влияет на план запроса, в то время как различия во времени не существенны.

5 Выбор порядка соединения Сравнение времени выполнения

5.2 задание 19.10 - 19.11

Перед выполнением запросов была выполнена команда set statistics time on для измерения времени выполнения следующих запросов.

Пример 19.10 Соединение таблиц методом вложенного цикла.

Метод вложенного цикла основан на применении полного перебора. То есть для каждой строки внешней таблицы извлекается и сравнивается каждая строка внутренней таблицы

Код запроса

|  |
| --- |
| SET SHOWPLAN\_TEXT ON;  go  SELECT \* FROM Заезд  JOIN МногоДанных ON Заезд.id\_Заезд = МногоДанных.idЗаезд  WHERE idЗаезд = 82082  go  SET SHOWPLAN\_TEXT OFF; |

Текстовый план запроса

|  |
| --- |
| |--Nested Loops(Inner Join)  |--Clustered Index Seek(OBJECT:([Hotel].[dbo].[Заезд].[PK\_\_Заезд\_\_D05F06D1EA809C2B]), SEEK:([Hotel].[dbo].[Заезд].[id\_Заезд]=(82082)) ORDERED FORWARD)  |--Clustered Index Scan(OBJECT:([Hotel].[dbo].[Статистика].[PK\_Статистика]), WHERE:([Hotel].[dbo].[Статистика].[idЗаезд]=(82082))) |

Время выполнения запроса

|  |
| --- |
| SQL Server parse and compile time:  CPU time = 0 ms, elapsed time = 1 ms. |

Пример 19.11 Запрос с использованием метода соединения хешированием.

Метод соединения хешированием обычно применяется при отсутствии индексов для столбцов соединения. Этот метод хорошо подходит для незапланированных запросов, для которых не предполагается наличие индексов.

Код запроса

|  |
| --- |
| SET SHOWPLAN\_TEXT ON;  go  SELECT \* FROM Заезд  JOIN МногоДанных ON Заезд.id\_Заезд = МногоДанных.idЗаезд  go  SET SHOWPLAN\_TEXT OFF; |

Текстовый план запроса

|  |
| --- |
| |--Hash Match(Inner Join, HASH:([Hotel].[dbo].[Заезд].[id\_Заезд])=([Hotel].[dbo].[Статистика].[idЗаезд]))  |--Clustered Index Scan(OBJECT:([Hotel].[dbo].[Заезд].[PK\_\_Заезд\_\_D05F06D1EA809C2B]))  |--Clustered Index Scan(OBJECT:([Hotel].[dbo].[Статистика].[PK\_Статистика])) |

Время выполнения запроса

|  |
| --- |
| SQL Server parse and compile time:  CPU time = 0 ms, elapsed time = 1 ms. |

6 динамические представления

6.1 задания 19.12 – 19.13

Пример 19.12 Применение представления sys.dm\_exec\_query\_optimizer\_info.

Представление sys.dm\_exec\_query\_optimizer\_info возвращает подробную статистику о работе оптимизатора. Это представление можно использовать при настройке рабочей нагрузки, чтобы определить проблемы оптимизации запросов и возможные ее улучшения.

Представление sys.dm\_exec\_query\_optimizer\_info содержит три столбца:

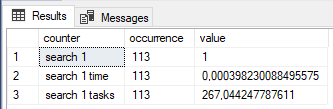
1. counter – имя события оптимизатора;
2. occurrence – общее количество проявлений этих событий;
3. value – дополнительная информация о событиях.

По причине своей сложности процесс оптимизации разбивается на три этапа:

1. планы непараллельного выполнения;
2. планы непараллельного и параллельного выполнения;
3. планы параллельного выполнения.

Код запроса

|  |
| --- |
| SELECT counter, occurrence,  value FROM sys.dm\_exec\_query\_optimizer\_info  WHERE value IS NOT NULL AND counter LIKE 'search 1%'; |



Результат запроса

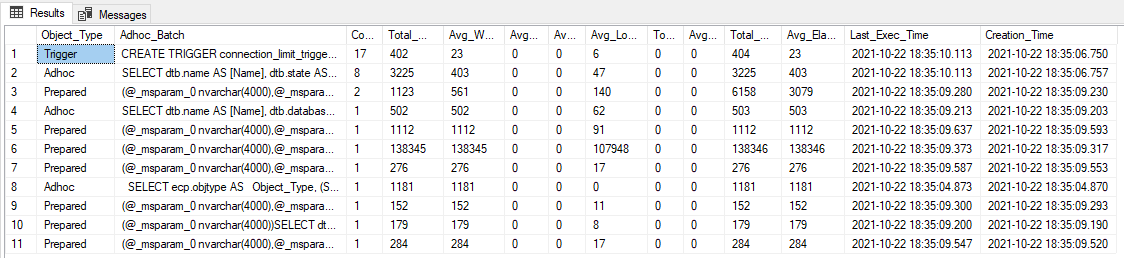
Пример 19.13 Применение представления sys.dm\_exec\_query\_stats

Представление sys.dm\_exec\_query\_stats возвращает суммарную статистику производительности для кэшированных планов выполнения запросов. Это представление содержит одну строку для каждой инструкции запроса в кэшированном плане, а время жизни строк связано с самим планом.

Запрос в примере 19.13 соединяет представления sys.dm\_exec\_ query\_stats и sys.dm\_exec\_cached\_plans и возвращает все кэшированные планы выполнения, упорядоченные по количеству их выполнений.

Код запроса

|  |
| --- |
| SELECT ecp.objtype AS  Object\_Type, (SELECT t. text FROM sys.dm\_exec\_sql\_text(qs.sql\_handle) AS t)  AS Adhoc\_Batch, qs.execution\_count  AS Counts, qs.total\_worker\_time  AS Total\_Worker\_Time, (qs.total\_worker\_time / qs.execution\_count)  AS Avg\_Worker\_Time, (qs.total\_physical\_reads / qs.execution\_count)  AS Avg\_Physical\_Reads, (qs.total\_logical\_writes / qs.execution\_count)  AS Avg\_Logical\_Writes, (qs.total\_logical\_reads / qs.execution\_count)  AS Avg\_Logical\_Reads, qs.total\_clr\_time  AS Total\_CLR\_Time, (qs.total\_clr\_time / qs.execution\_count)  AS Avg\_CLR\_Time, qs.total\_elapsed\_time  AS Total\_Elapsed\_Time, (qs.total\_elapsed\_time / qs.execution\_count)  AS Avg\_Elapsed\_Time, qs.last\_execution\_time  AS Last\_Exec\_Time, qs.creation\_time  AS Creation\_Time FROM sys.dm\_exec\_query\_stats  AS qs JOIN sys.dm\_exec\_cached\_plans ecp ON qs.plan\_handle = ecp.plan\_handle  ORDER BY Counts DESC; |



Результат запроса

7 Подсказки оптимизации Сравнение времени выполнения

задания 19.14 - 19.20

Перед выполнением запросов была выполнена команда set statistics time on для измерения времени выполнения следующих запросов.

Пример 19.14 Принуждение оптимизатора к использованию индекса.

содержит подсказку INDEX, которая заставляет оптимизатор запросов использовать индекс PK\_Статистика.

Код запроса

|  |
| --- |
| SET SHOWPLAN\_TEXT ON;  go  SELECT \* FROM Статистика  WITH (INDEX(PK\_Статистика))  WHERE id = 40010;  go  SET SHOWPLAN\_TEXT OFF; |

Текстовый план запроса

|  |
| --- |
| |--Clustered Index Seek(OBJECT:([Hotel].[dbo].[Статистика].[PK\_Статистика]), SEEK:([Hotel].[dbo].[Статистика].[id]=(40010)) ORDERED FORWARD) |

Время выполнения запроса

|  |
| --- |
| SQL Server parse and compile time:  CPU time = 0 ms, elapsed time = 0 ms. |

Пример 19.15 Принуждение оптимизатора не использовать некластеризованные индексы.

Подсказка INDEX(0), заставляет оптимизатор не использовать никакие существующие некластеризованные индексы.

Код запроса

|  |
| --- |
| SET SHOWPLAN\_TEXT ON;  go  SELECT \* FROM Статистика  WITH (INDEX(0))  WHERE id = 40010;  go  SET SHOWPLAN\_TEXT OFF; |

Текстовый план запроса

|  |
| --- |
| |--Clustered Index Scan(OBJECT:([Hotel].[dbo].[Статистика].[PK\_Статистика]), WHERE:([Hotel].[dbo].[Статистика].[id]=(40010))) |

Время выполнения запроса

|  |
| --- |
| SQL Server parse and compile time:  CPU time = 0 ms, elapsed time = 0 ms. |

Пример 19.16 Использовать больше таблиц Использование подсказки FORCE ORDER.

Подсказка FORCE ORDER заставляет оптимизатор соединять таблицы в том порядке, в котором они указаны в запросе.

Код запроса с

|  |
| --- |
| SET SHOWPLAN\_TEXT ON;  go  SELECT \* FROM Заезд  JOIN Статистика ON Заезд.id\_Заезд = Статистика.idЗаезд  JOIN Номер on idНомера = Номер\_idНомера  JOIN Размещение on id\_Размещение = Размещение\_id\_Размещение  OPTION(FORCE ORDER);  go  SET SHOWPLAN\_TEXT OFF; |

Текстовый план запроса без опции FORCE ORDER

|  |
| --- |
| |--Hash Match(Inner Join, HASH:([Hotel].[dbo].[Размещение].[id\_Размещение])=([Hotel].[dbo].[Номер].[Размещение\_id\_Размещение]))  |--Clustered Index Scan(OBJECT:([Hotel].[dbo].[Размещение].[PK\_\_Размещен\_\_7BD159D8EC88B28F]))  |--Hash Match(Inner Join, HASH:([Hotel].[dbo].[Номер].[idНомера])=([Hotel].[dbo].[Заезд].[Номер\_idНомера]))  |--Clustered Index Scan(OBJECT:([Hotel].[dbo].[Номер].[PK\_\_Номер\_\_361744934FCB1498]))  |--Hash Match(Inner Join, HASH:([Hotel].[dbo].[Заезд].[id\_Заезд])=([Hotel].[dbo].[Статистика].[idЗаезд]))  |--Clustered Index Scan(OBJECT:([Hotel].[dbo].[Заезд].[PK\_\_Заезд\_\_D05F06D1EA809C2B]))  |--Clustered Index Scan(OBJECT:([Hotel].[dbo].[Статистика].[PK\_Статистика])) |

Время выполнения запроса без опции FORCE ORDER

|  |
| --- |
| SQL Server parse and compile time:  CPU time = 0 ms, elapsed time = 5 ms. |

Текстовый план запроса с опцией FORCE ORDER

|  |
| --- |
| |--Merge Join(Inner Join, MANY-TO-MANY MERGE:([Hotel].[dbo].[Номер].[Размещение\_id\_Размещение])=([Hotel].[dbo].[Размещение].[id\_Размещение]), RESIDUAL:([Hotel].[dbo].[Номер].[Размещение\_id\_Размещение]=[Hotel].[dbo].[Размещение].[id\_Размещение]))  |--Sort(ORDER BY:([Hotel].[dbo].[Номер].[Размещение\_id\_Размещение] ASC))  | |--Merge Join(Inner Join, MANY-TO-MANY MERGE:([Hotel].[dbo].[Заезд].[Номер\_idНомера])=([Hotel].[dbo].[Номер].[idНомера]), RESIDUAL:([Hotel].[dbo].[Номер].[idНомера]=[Hotel].[dbo].[Заезд].[Номер\_idНомера]))  | |--Sort(ORDER BY:([Hotel].[dbo].[Заезд].[Номер\_idНомера] ASC))  | | |--Hash Match(Inner Join, HASH:([Hotel].[dbo].[Заезд].[id\_Заезд])=([Hotel].[dbo].[Статистика].[idЗаезд]))  | | |--Clustered Index Scan(OBJECT:([Hotel].[dbo].[Заезд].[PK\_\_Заезд\_\_D05F06D1EA809C2B]))  | | |--Clustered Index Scan(OBJECT:([Hotel].[dbo].[Статистика].[PK\_Статистика]))  | |--Clustered Index Scan(OBJECT:([Hotel].[dbo].[Номер].[PK\_\_Номер\_\_361744934FCB1498]), ORDERED FORWARD)  |--Clustered Index Scan(OBJECT:([Hotel].[dbo].[Размещение].[PK\_\_Размещен\_\_7BD159D8EC88B28F]), ORDERED FORWARD) |

Время выполнения запроса с опцией FORCE ORDER

|  |
| --- |
| SQL Server parse and compile time:  CPU time = 0 ms, elapsed time = 3 ms. |

Пример 19.17 Использование подсказки слияния MERGE.

Подсказка запроса MERGE заставляет оптимизатор использовать метод соединения слиянием.

Код запроса

|  |
| --- |
| SET SHOWPLAN\_TEXT ON;  go  SELECT \* FROM Заезд  JOIN Статистика ON Заезд.id\_Заезд = Статистика.idЗаезд  OPTION(MERGE JOIN);  go  SET SHOWPLAN\_TEXT OFF; |

Текстовый план запроса

|  |
| --- |
| |--Merge Join(Inner Join, MERGE:([Hotel].[dbo].[Заезд].[id\_Заезд])=([Hotel].[dbo].[Статистика].[idЗаезд]),RESIDUAL:([Hotel].[dbo].[Статистика].[idЗаезд]=[Hotel].[dbo].[Заезд].[id\_Заезд]))  |--Clustered Index Scan(OBJECT:([Hotel].[dbo].[Заезд].[PK\_\_Заезд\_\_D05F06D1EA809C2B]), ORDERED FORWARD)  |--Sort(ORDER BY:([Hotel].[dbo].[Статистика].[idЗаезд] ASC))  |--Clustered Index Scan(OBJECT:([Hotel].[dbo].[Статистика].[PK\_Статистика])) |

Время выполнения запроса

|  |
| --- |
| SQL Server parse and compile time:  CPU time = 0 ms, elapsed time = 1 ms. |

Пример 19.18 Использование подсказки слияния MERGE в предложении FROM.

Аналогичен предыдущему примеру за исключением места, в котором используется подсказка MERGE.

Код запроса

|  |
| --- |
| SELECT \* FROM Заезд  INNER MERGE JOIN Статистика ON Заезд.id\_Заезд = Статистика.idЗаезд |

Текстовый план запроса

|  |
| --- |
| |--Merge Join(Inner Join, MERGE:([Hotel].[dbo].[Заезд].[id\_Заезд])=([Hotel].[dbo].[Статистика].[idЗаезд]), RESIDUAL:([Hotel].[dbo].[Статистика].[idЗаезд]=[Hotel].[dbo].[Заезд].[id\_Заезд]))  |--Clustered Index Scan(OBJECT:([Hotel].[dbo].[Заезд].[PK\_\_Заезд\_\_D05F06D1EA809C2B]), ORDERED FORWARD)  |--Sort(ORDER BY:([Hotel].[dbo].[Статистика].[idЗаезд] ASC))  |--Clustered Index Scan(OBJECT:([Hotel].[dbo].[Статистика].[PK\_Статистика])) |

Время выполнения запроса

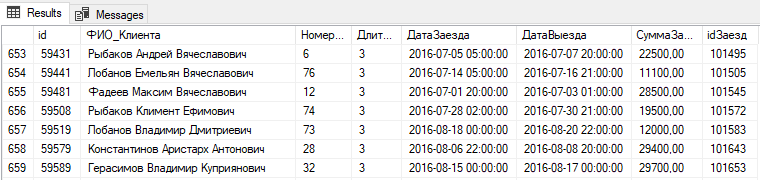
|  |
| --- |
| SQL Server parse and compile time:  CPU time = 0 ms, elapsed time = 1 ms. |

Пример 19.19 Использование подсказки запроса OPTIMIZE FOR.

Подсказка OPTIMIZE FOR заставляет оптимизатор запросов использовать определенное значение для локальной переменной при компилировании и оптимизации запроса.

Код запроса

|  |
| --- |
| DECLARE @cost int SET @cost = 3  SELECT \* FROM Статистика WHERE ДлительностьПроживания = @cost  OPTION (OPTIMIZE FOR (@cost = 6)); |



Результат запроса

Время выполнения запроса

|  |
| --- |
| SQL Server parse and compile time:  CPU time = 0 ms, elapsed time = 1 ms. |

Пример 19.20 Создание структуры плана.

Cтруктуры планов позволяют использовать определенную подсказку оптимизации, не изменяя синтаксис инструкции SELECT.

Код запроса

|  |
| --- |
| go  SET SHOWPLAN\_TEXT ON;  SELECT \* FROM Заезд JOIN Статистика ON Заезд.id\_Заезд = Статистика.idЗаезд  go  SET SHOWPLAN\_TEXT OFF; |

Текстовый план запроса

|  |
| --- |
| |--Hash Match(Inner Join, HASH:([Hotel].[dbo].[Заезд].[id\_Заезд])=([Hotel].[dbo].[Статистика].[idЗаезд]))  |--Clustered Index Scan(OBJECT:([Hotel].[dbo].[Заезд].[PK\_\_Заезд\_\_D05F06D1EA809C2B]))  |--Clustered Index Scan(OBJECT:([Hotel].[dbo].[Статистика].[PK\_Статистика])) |

Код запроса

|  |
| --- |
| sp\_create\_plan\_guide @name = N'Example\_19\_15',  @stmt = N'SELECT \* FROM Заезд JOIN Статистика ON Заезд.id\_Заезд = Статистика.idЗаезд',  @type = N'SQL',  @module\_or\_batch = NULL,  @params = NULL,  @hints = N'OPTION (MERGE JOIN)'  go  SET SHOWPLAN\_TEXT ON;  go  SELECT \* FROM Заезд JOIN Статистика ON Заезд.id\_Заезд = Статистика.idЗаезд  go  SET SHOWPLAN\_TEXT OFF; |

Текстовый план запроса

|  |
| --- |
| |--Merge Join(Inner Join, MERGE:([Hotel].[dbo].[Заезд].[id\_Заезд])=([Hotel].[dbo].[Статистика].[idЗаезд]), RESIDUAL:([Hotel].[dbo].[Статистика].[idЗаезд]=[Hotel].[dbo].[Заезд].[id\_Заезд]))  |--Clustered Index Scan(OBJECT:([Hotel].[dbo].[Заезд].[PK\_\_Заезд\_\_D05F06D1EA809C2B]), ORDERED FORWARD)  |--Sort(ORDER BY:([Hotel].[dbo].[Статистика].[idЗаезд] ASC))  |--Clustered Index Scan(OBJECT:([Hotel].[dbo].[Статистика].[PK\_Статистика])) |

8 Настройка производительности

задания 20.1 - 20.13

Пример 20.1 Использование представления каталога sys.sysprocesses.

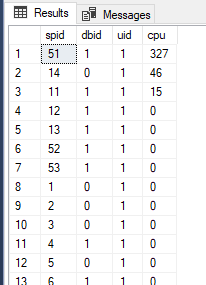
Представление каталога sys.sysprocesses может быть полезным для определения процессов, которые используют больше всего процессорного времени.

Наиболее важными столбцами этого представления являются столбцы:

1. spid – идентификатор сессии;
2. dbid – идентификатор текущей базы данных;
3. uid – идентификатор пользователя, выполняющего текущую команду;
4. cpu – общее время центрального процессора, затраченного на данный процесс.

Код запроса

|  |
| --- |
| USE master;  SELECT spid, dbid, uid, cpu  FROM master.dbo.sysprocesses order by cpu DESC; |



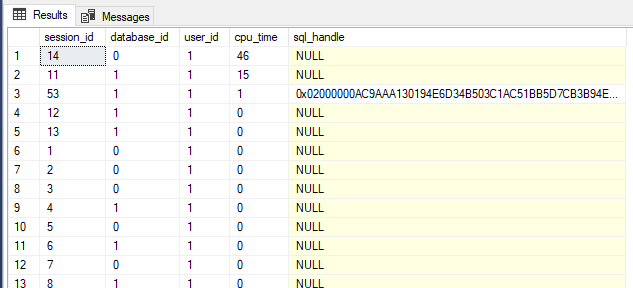
Результат запроса

Пример 20.2 Использование ДАП sys.dm\_exec\_requests.

Динамическое административное представление sys.dm\_exec\_requests предоставляет ту же информацию, что и представление каталога sys.sysprocesses, но его столбцы называются по-другому.

Код запроса

|  |
| --- |
| SELECT session\_id, database\_id, user\_id, cpu\_time, sql\_handle  FROM sys.dm\_exec\_requests order by cpu\_time DESC; |



Результат запроса

Пример 20.3 Использование динамического административного представления sys.dm\_exec\_query\_stats.

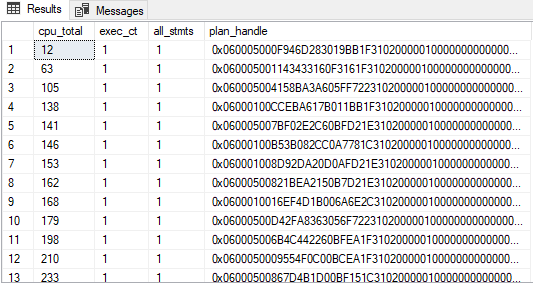
Используется для отображения информации о кэшированных инструкциях Transact-SQL и хранимых процедурах, которые используют больше всего времени центрального процессора.

В столбце total\_worker\_time отображается общее время центрального процессора, использованное на выполнение кэшированных SQL-инструкций и хранимых процедур с момента их компиляции.

В столбце execution\_count отображается количество времени, которое кэшированных планы будут выполняться с момента их последней компиляции.

Код запроса

|  |
| --- |
| SELECT TOP 20 SUM(total\_worker\_time)  AS cpu\_total, SUM(execution\_count)  AS exec\_ct, COUNT(\*)  AS all\_stmts, plan\_handle  FROM sys.dm\_exec\_query\_stats  GROUP BY plan\_handle ORDER BY cpu\_total; |



Результат запроса

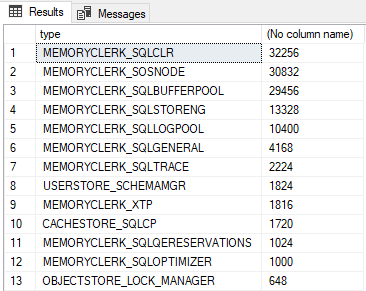
Пример 20.4 Использование представления sys.dm\_os\_memory\_clerks.

Возвращает набор всех служб памяти, которые являются активными в текущем экземпляре сервера. С помощью этого представления можно получить распределение памяти для различных типов памяти.

Столбец type представления sys.dm\_os\_memory\_clerks содержит описание типа службы памяти, а в столбце pages\_kb указывается объем памяти, выделенный с помощью распределителя одиночных страниц узла памяти.

Код запроса

|  |
| --- |
| SELECT type, SUM(pages\_kb)  FROM sys.dm\_os\_memory\_clerks  WHERE pages\_kb != 0  GROUP BY type ORDER BY 2 DESC; |



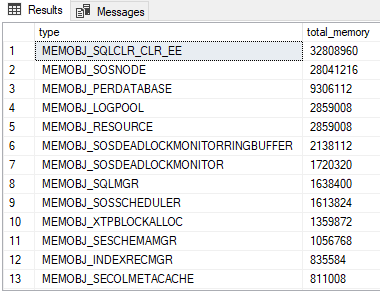
Результат запроса

Пример 20.5 Использование динамического административного представления sys.dm\_os\_memory\_objects.

Возвращает объекты памяти, которые в настоящее время распределены системой баз данных. Используется для анализа использования памяти и для определения возможного недостатка памяти.

Код запроса

|  |
| --- |
| SELECT type, SUM(pages\_in\_bytes)  AS total\_memory  FROM sys.dm\_os\_memory\_objects  GROUP BY type ORDER BY total\_memory DESC; |



Результат запроса

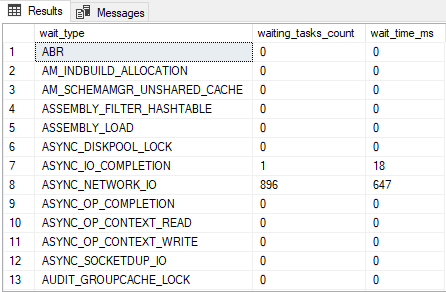
Пример 20.6 Использование динамического административного представления sys.dm\_os\_wait\_stats.

Возвращает информацию об ожиданиях в выполняемых потоках. Это представление можно использовать для диагностики проблем производительности как всего сервера Database Engine, так и отдельных запросов и пакетов.

Столбец wait\_type содержит названия типов ожидания, а столбец waiting\_tasks\_count содержит количество ожиданий соответствующего типа.

Код запроса

|  |
| --- |
| SELECT wait\_type, waiting\_tasks\_count, wait\_time\_ms  FROM sys.dm\_os\_wait\_stats  ORDER BY wait\_type; |



Результат запроса

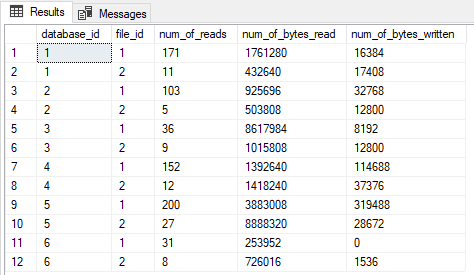
Пример 20.7 Использование динамического административного представления sys.dm\_io\_virtual\_file\_stats.

Возвращает статистику ввода/вывода для файлов данных и журнала.

В database\_id, указывается однозначный идентификационный номер базы данных, а в file\_id – идентификатор файла. Когда указано значение NULL, тогда возвращаются все базы данных.

Код запроса

|  |
| --- |
| SELECT database\_id, file\_id, num\_of\_reads,  num\_of\_bytes\_read, num\_of\_bytes\_written  FROM sys.dm\_io\_virtual\_file\_stats (NULL, NULL); |



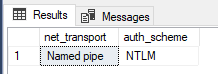
Результат запроса

Пример 20.8 Отображение сетевого протокола и механизма проверки подлинности.

Данный пример возвращает основную информацию о текущем соединении: сетевой транспортный протокол и механизм аутентификации из представления sys.dm\_exec\_connections.

Код запроса

|  |
| --- |
| SELECT net\_transport, auth\_scheme  FROM sys.dm\_exec\_connections  WHERE session\_id=@@SPID; |



Результат запроса

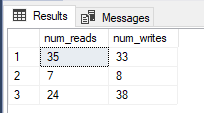
Пример 20.9 Отображение количества пакетов чтения и записи.

В данном примере показано использование двух важных столбцов представления sys.dm\_exec\_connection:

1. num\_reads – отображает количество прочитанных пакетов в текущем соединении;
2. num\_writes – количество записанных пакетов.

Код запроса

|  |
| --- |
| SELECT num\_reads, num\_writes  FROM sys.dm\_exec\_connections; |

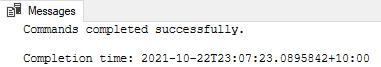


Результат запроса

Пример 20.10 Создание двух новых таблиц для базы данных sample.

Код запроса

|  |
| --- |
| USE sample;  CREATE TABLE orders (  orderid INTEGER NOT NULL,  orderdate DATE,  shippeddate DATE,  freight money);  CREATE TABLE order\_details (  productid INTEGER NOT NULL,  orderid INTEGER NOT NULL,  unitprice money,  quantity INTEGER); |

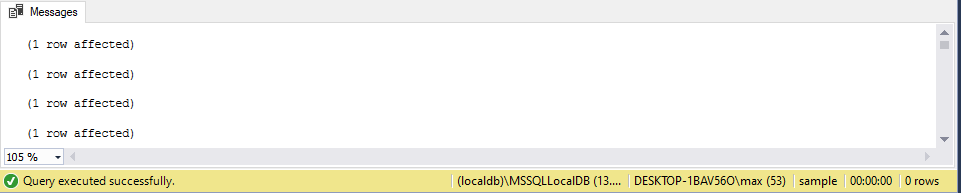


Результат запроса

Пример 20.11 Вставка новых строк в таблицу orders

Код запроса

|  |
| --- |
| USE sample;  declare @i int, @order\_id integer  declare @orderdate datetime  declare @shipped\_date datetime  declare @freight money  set @i = 1  set @orderdate = getdate()  set @shipped\_date = getdate()  set @freight = 100.00 while @i < 3001  begin  insert into orders (orderid, orderdate, shippeddate, freight)  values(@i, @orderdate, @shipped\_date, @freight)  set @i = @i+1  end |

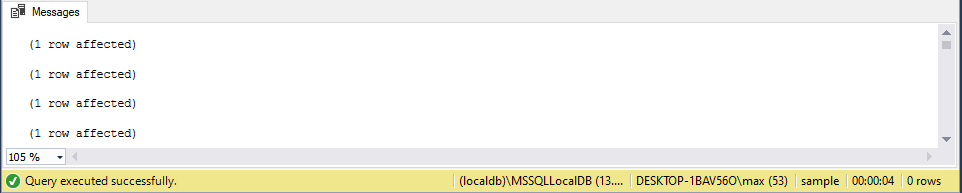


Результат запроса

Пример 20.12 Вставка новых строк в таблицу order\_details

Код запроса

|  |
| --- |
| USE sample;  declare @i int, @j int  set @i = 3000  set @j = 10  while @j > 0 begin  if @i > 0 begin  insert into  order\_details (productid, orderid, quantity)  values (@i, @j, 5)  set @i = @i - 1  end else begin  set @j = @j - 1  set @i = 3000  end  end  go  update order\_details  set quantity = 3  where productid in (1511, 2678) |



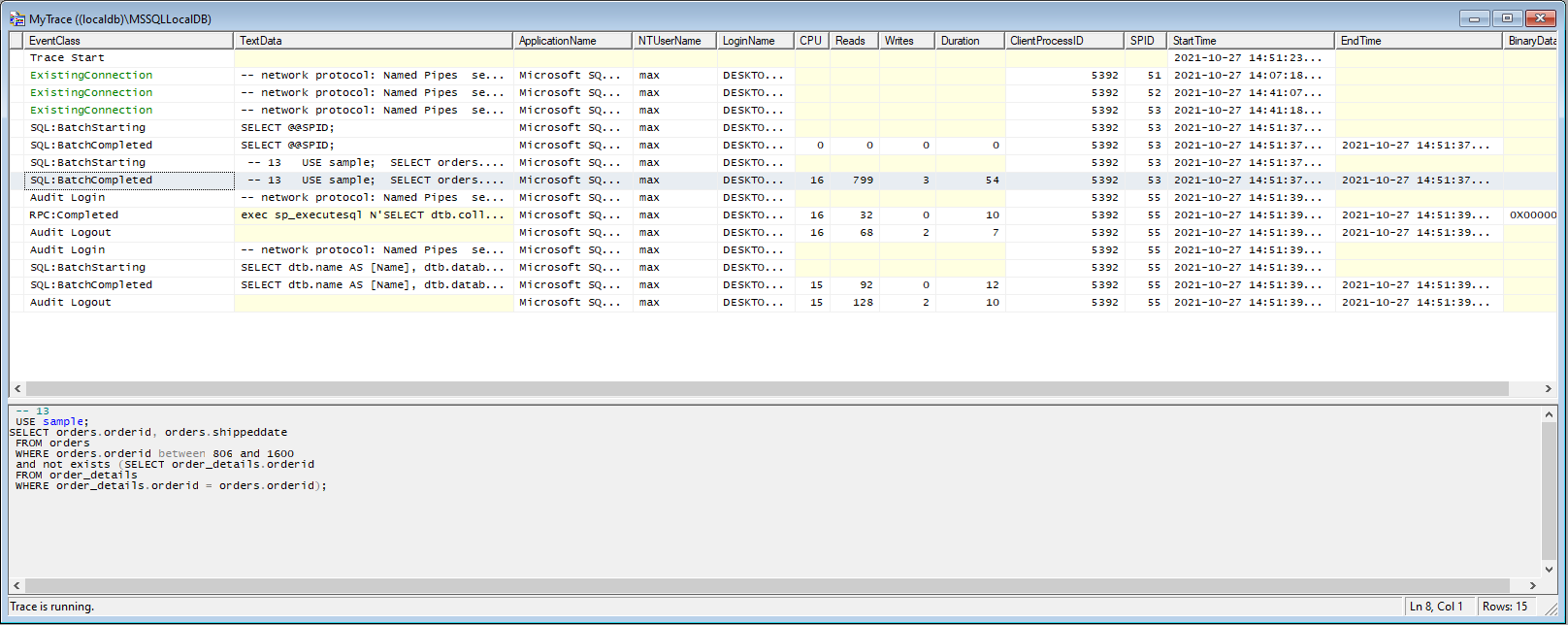
Результат запроса

Пример 20.13 Запрос для трассировки

С помощью приложения SQL Server Profiler можно записывать в файл трассировки информацию об исследуемой рабочей нагрузке. После этого помощник Database Engine Tuning Advisor может прочитать этот файл и порекомендовать создать для данной рабочей нагрузки несколько физических объектов, таких как индексы, индексированные представления или схема секционирования.

Код запроса

|  |
| --- |
| USE sample;  SELECT orders.orderid, orders.shippeddate  FROM orders  WHERE orders.orderid between 806 and 1600  and not exists (SELECT order\_details.orderid  FROM order\_details  WHERE order\_details.orderid = orders.orderid); |



Результат трассировки