Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Комсомольский-на-Амуре государственный университет»

Факультет компьютерных технологий

Кафедра «ПУРИС»

РАСЧЕТНО-ГРАФИЧЕСКАЯ РАБОТА

по дисциплине «Администрирование баз данных»

Студент группы 8ВТб-1 А.С. Яковлев

Преподаватель А.Н. Петрова

2021

**Задание**

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

2 Отобразить план выполнения запроса.

3 Проанализировать 6 запросов с выражением в условии отбора.

4 Проанализировать влияние индексов и статистики.

5 Рассмотреть порядки соединения.

6 Рассмотреть динамические представления.

7 Рассмотреть подсказки оптимизации.

8 Произвести настройку производительности.

**Содержание**

[Введение 4](#_Toc89330264)

[1 Таблица для тестирования запросов 5](#_Toc89330265)

[2 Отображение плана запроса 7](#_Toc89330266)

[3 Анализ запросов 10](#_Toc89330267)

[4 Индексы и статистика 15](#_Toc89330268)

[5 Выбор порядка соединения 24](#_Toc89330269)

[6 Динамические представления 26](#_Toc89330270)

[7 Подсказки оптимизации 28](#_Toc89330271)

[8 Настройка производительности 33](#_Toc89330272)

[Заключение 42](#_Toc89330273)

[Список использованных источников 43](#_Toc89330274)

# Введение

При выполнении запроса компонентом Database Engine обычно возникает вопрос получения доступа и обработки требуемых для запроса данных с максимальной эффективностью. Компонент системы баз данных, ответственный за это, называется оптимизатором запросов [1].

Задача оптимизатора запросов состоит в рассмотрении различных возможных стратегий для выборки данных для конкретного запроса и выбора наиболее эффективной стратегии. Выбранная стратегия называется планом выполнения запроса. Оптимизатор принимает решения, принимая во внимание такие факторы, как размер таблиц, к которым направлен запрос, существующие индексы и логические операторы (AND, OR или NOT), используемые в предложении WHERE. Совместно эти факторы называются статистическими данными.

В данной работе будет создана таблица «Статистика» в базе данных «Movie», которая содержит большое количество записей – более десяти тысячей.

Также будет пройдена тема анализа запроса, в которой оптимизатор пытается ускорить время выполнения запроса. Также в работе будет разобран план выполнения запроса, изучены индексы и статистики для оптимизации времени выполнения запроса. Будут разобраны порядки соединения – метод вложенного цикла и соединения хэшированием, динамические представления, подсказки оптимизации и настройка производительности.

# 1 Таблица для тестирования запросов

Для того, чтобы можно было увидеть какую-либо разницу при оптимизации запросов, была создана таблица «Статистика».

В листинге 1.1 представлен код, который использовался при создании таблицы «Статистика», в которой находится более десяти тысячи записей.

Листинг 1.1 – Код для создания таблицы «Статистика»

|  |
| --- |
| IF OBJECT\_ID ( '[add\_stat]', 'P' ) IS NOT NULL  DROP PROCEDURE add\_stat;  GO  CREATE PROCEDURE [dbo].add\_stat  @howMuch int,  @costTicket int  AS  BEGIN  SET NOCOUNT ON;  declare @idFilm int;  declare @countTickets int;  declare @money money;  declare @dateTime datetime;  set @dateTime = N'2010-01-01 09:00:00';  DECLARE @dTime time;  DECLARE @newTime time;  set @newTime = N'00:00:00';  DECLARE @counter INT;  SET @counter = 0;  WHILE @counter < @howMuch  BEGIN  set @idFilm = ABS(CHECKSUM(NEWID()) % (16 - 6 - 1)) + 6; -- от 6 до 16  set @countTickets = ABS(CHECKSUM(NEWID()) % (260 - 100 - 1)) + 100; -- от 100 до 260  set @money = @countTickets \* @costTicket;  SELECT @dTime = cast(@dateTime as time)  IF @dTime <> @newTime set @dateTime = DATEADD(HOUR, 3, @dateTime);  ELSE set @dateTime = DATEADD(HOUR, 9, @dateTime);  insert [dbo].Статистика(Фильм\_idФильм,Дата\_время,[Кол-во\_проданных\_билетов],Заработок)  values (@idFilm,@dateTime,@countTickets,@money);  set @counter = @counter + 1  END;  END |

На рисунках 1.1 и 1.2 представлена структура и содержимое таблицы «Статистика».

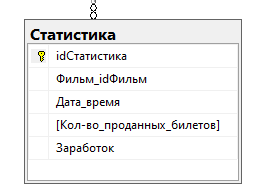


Рисунок 1.1 – Структура таблицы «Статистика»

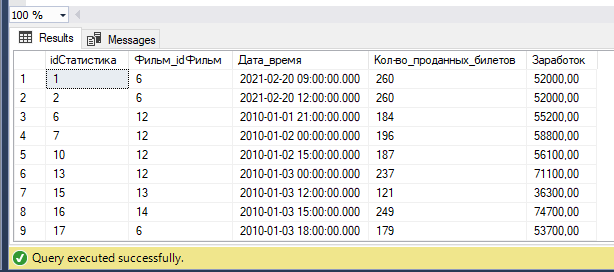


Рисунок 1.2 – Содержимое таблицы «Статистика»

# 2 Отображение плана запроса

Термин текстовый план выполнения означает, что план выполнения запроса отображается построчно в текстовом формате.

Для отображения текстового плана выполнения используются следующие опции инструкции SET:

- SHOWPLAN\_TEXT;

- SHOWPLAN\_ALL.

Отображение текстового плана запроса посредством опции SHOWPLAN\_TEXT представлено в листинге 2.1.

Листинг 2.1 – Код для реализации опции SHOWPLAN\_TEXT

|  |
| --- |
| SET SHOWPLAN\_TEXT ON;  GO  USE Movie;  SELECT \* FROM Фильм F JOIN Статистика S  ON F.idФильм = S.Фильм\_idФильм  AND F.idФильм = 6;  GO  SET SHOWPLAN\_TEXT OFF; |

На рисунке 2.1 представлено отображение текстового плана запроса посредством опции SHOWPLAN\_TEXT.

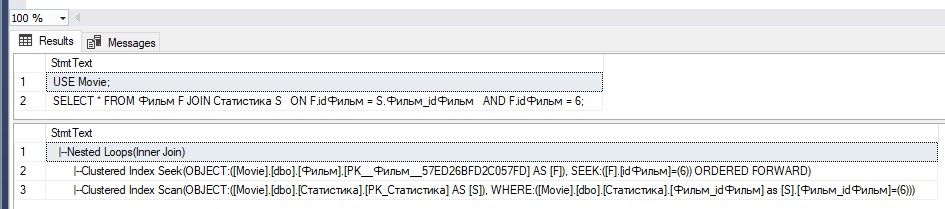


Рисунок 2.1 – Отображение текстового плана запроса

Термин план выполнения XML означает, что план выполнения запроса отображается в формате XML.

Для отображения планов выполнения в формате XML используются следующие две опции инструкции SET:

- SHOWPLAN\_XML;

- STATISTICS XML.

Отображение плана выполнения посредством опции SHOWPLAN\_XML представлено в листинге 2.2.

Листинг 2.2 – Код для реализации опции SHOWPLAN\_ XML

|  |
| --- |
| SET SHOWPLAN\_XML ON;  GO  USE Movie;  SELECT \* FROM Фильм F JOIN Статистика S  ON F.idФильм = S.Фильм\_idФильм  AND F.idФильм = 6;  GO  SET SHOWPLAN\_XML OFF; |

На рисунке 2.2 представлено отображение плана выполнения посредством опции SHOWPLAN\_XML.



Рисунок 2.2 – Отображение плана выполнения

Использование опции STATISTICS PROFILE в инструкции SET, которая содержит вывод инструкций SET STATISTICS IO и SET SHOWPLAN\_ALL представлено в листинге 2.3.

Листинг 2.3 – Код для реализации опции STATISTICS PROFILE

|  |
| --- |
| SET STATISTICS PROFILE ON;  GO  USE Movie;  SELECT \* FROM Фильм F JOIN Статистика S  ON F.idФильм = S.Фильм\_idФильм  AND F.idФильм = 6;  GO  SET STATISTICS PROFILE OFF; |

На рисунке 2.3 представлено использование опции STATISTICS PROFILE в инструкции SET.

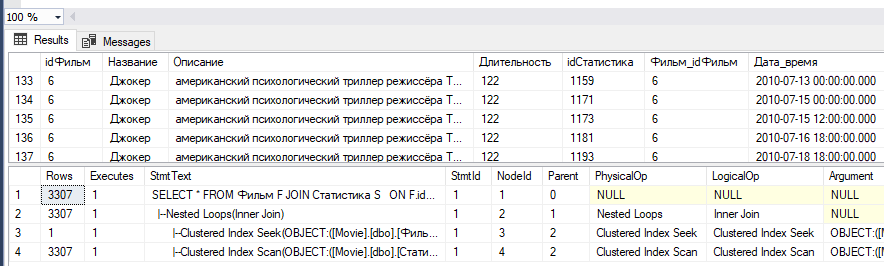


Рисунок 2.3 – Использование опции STATISTICS PROFILE

# 3 Анализ запросов

В процессе анализа запроса оптимизатор исследует его на наличие аргументов поиска, использование оператора OR и существование критериев соединения, в приведенном порядке. Аргумент поиска – это часть запроса, которая ограничивает промежуточный результирующий набор запроса. Основным назначением аргументов поиска является то, что они позволяют использовать существующие индексы применительно к конкретному выражению.

Для вывода времени в миллисекундах были использованы инструкции, представленные в листинге 3.1.

Листинг 3.1 – Инструкции для вывода времени

|  |
| --- |
| set statistics time on  set statistics io on |

В листинге 3.2 представлены два запроса, которые выводят одинаковый результат, но при этом построены по-разному.

Первый запрос является неправильным, так как содержит в себе оператор <>. Второй запрос является правильным, так как содержит в себе оператор >= и/или < вместо оператора <>.

Листинг 3.2 – Операторы сравнения

|  |
| --- |
| SET STATISTICS PROFILE ON;  GO  SELECT \* FROM Статистика WHERE Фильм\_idФильм <> 12 AND Фильм\_idФильм <> 13  SELECT \* FROM Статистика WHERE Фильм\_idФильм < 12 OR Фильм\_idФильм > 13  GO  SET STATISTICS PROFILE OFF; |

В листингах 3.3 и 3.4 представлены планы выполнения запросов.

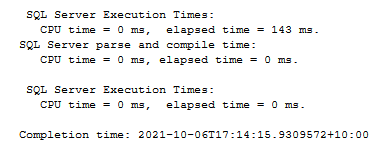
Листинг 3.3 – План выполнения запроса с неправильным оператором

|  |
| --- |
| |--Clustered Index Scan(  OBJECT:([Movie].[dbo].[Статистика].[PK\_Статистика]), WHERE:([Movie].[dbo].[Статистика].[Фильм\_idФильм]<>(12) AND [Movie].[dbo].[Статистика].[Фильм\_idФильм]<>(13))) |

Листинг 3.4 – План выполнения запроса с правильным оператором

|  |
| --- |
| |--Clustered Index Scan(  OBJECT:([Movie].[dbo].[Статистика].[PK\_Статистика]), WHERE:([Movie].[dbo].[Статистика].[Фильм\_idФильм]<(12) OR [Movie].[dbo].[Статистика].[Фильм\_idФильм]>(13))) |

На рисунке 3.1 представлено время выполнения запросов. Неправильный запрос потратил на выполнение 143 мс, правильный – 123 мс.



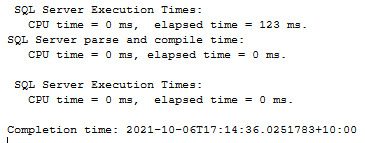


Рисунок 3.1 – Время выполнения запросов первого сравнения

В листинге 3.5 представлены два запроса, которые выводят одинаковый результат, но при этом построены по-разному.

Первый запрос является неправильным, так как сравнивает поле таблицы, умноженное на число с константой. Второй запрос является правильным, так как в нём сравнивается поле таблицы с выражением.

Листинг 3.5 – Вычисление выражения

|  |
| --- |
| SET STATISTICS PROFILE ON;  GO  SELECT \* FROM Статистика WHERE Фильм\_idФильм\*100 > 1000  SELECT \* FROM Статистика WHERE Фильм\_idФильм > 1000/100  GO  SET STATISTICS PROFILE OFF; |

В листингах 3.6 и 3.7 представлены планы выполнения запросов.

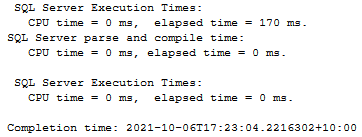
Листинг 3.6 – План выполнения запроса с неправильным выражением

|  |
| --- |
| |--Clustered Index Scan(  OBJECT:([Movie].[dbo].[Статистика].[PK\_Статистика]), WHERE:([Movie].[dbo].[Статистика].[Фильм\_idФильм]\*(100)>(1000))) |

Листинг 3.7 – План выполнения запроса с правильным выражением

|  |
| --- |
| |--Clustered Index Scan(  OBJECT:([Movie].[dbo].[Статистика].[PK\_Статистика]), WHERE:([Movie].[dbo].[Статистика].[Фильм\_idФильм]>[@1]/[@2])) |

На рисунке 3.2 представлено время выполнения запросов. Неправильный запрос потратил на выполнение 170 мс, правильный – 158 мс.



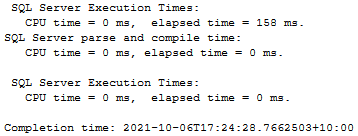


Рисунок 3.2 – Время выполнения запросов второго сравнения

В листинге 3.8 представлены два запроса, которые выводят одинаковый результат, но при этом построены по-разному.

Первый запрос является неправильным, так как содержит в себе предикат NOT IN. Второй запрос является правильным, так как не содержит в себе предикат NOT IN.

Листинг 3.8 – Предикат NOT IN

|  |
| --- |
| SET STATISTICS PROFILE ON;  GO  SELECT \* FROM Статистика WHERE Фильм\_idФильм NOT IN (12,13)  SELECT \* FROM Статистика WHERE Фильм\_idФильм < 12 OR Фильм\_idФильм > 13  GO  SET STATISTICS PROFILE OFF; |

В листингах 3.9 и 3.10 представлены планы выполнения запросов.

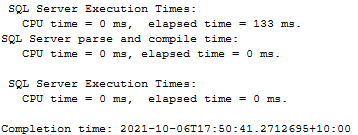
Листинг 3.9 – План выполнения запроса с предикатом NOT IN

|  |
| --- |
| |--Clustered Index Scan(OBJECT:([Movie].[dbo].[Статистика].[PK\_Статистика]), WHERE:([Movie].[dbo].[Статистика].[Фильм\_idФильм]<>(12) AND [Movie].[dbo].[Статистика].[Фильм\_idФильм]<>(13))) |

Листинг 3.10 – План выполнения запроса без предиката NOT IN

|  |
| --- |
| |--Clustered Index Scan(OBJECT:([Movie].[dbo].[Статистика].[PK\_Статистика]), WHERE:([Movie].[dbo].[Статистика].[Фильм\_idФильм]<(12) OR [Movie].[dbo].[Статистика].[Фильм\_idФильм]>(13))) |

На рисунке 3.3 представлено время выполнения запросов. Неправильный запрос потратил на выполнение 133 мс, правильный – 120 мс.



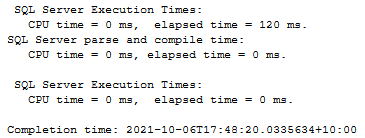


Рисунок 3.3 – Время выполнения запросов третьего сравнения

**Вывод**. Результаты по времени выполнению правильных и неправильных запросов отличаются между собой от 10 до 20 миллисекунд. Из этого можно сделать вывод, что технологии шагнули вперед и MSQL выполняет данные запросы одинаковы по затраченному времени, если не обращать внимания на погрешность.

# 4 Индексы и статистика

Запрос с высоким уровнем селективности – запрос на малое количество записей представлен в листинге 4.1. В листинге 4.2 представлен план выполнения этого запроса.

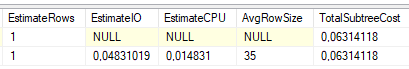
Листинг 4.1 – Запрос с высоким уровнем селективности

|  |
| --- |
| SET STATISTICS PROFILE ON;  GO  USE Movie;  SELECT \* FROM Статистика S  WHERE S.Фильм\_idФильм = 15  GO  SET STATISTICS PROFILE OFF; |

Листинг 4.2 – План выполнения запроса с высоким уровнем селективности

|  |
| --- |
| |--Clustered Index Scan(OBJECT:([Movie].[dbo].[Статистика].[PK\_Статистика] AS [S]), WHERE:([Movie].[dbo].[Статистика].[Фильм\_idФильм] as [S].[Фильм\_idФильм]=(15))) |

На рисунке 4.1 представлено время, затраченное на выполнения данного запроса.



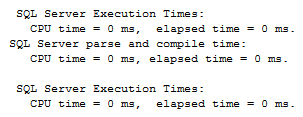


Рисунок 4.1 – Время выполнения запроса с

высоким уровнем селективности

Запрос с низким уровнем селективности – запрос на большое количество записей представлен в листинге 4.3. В листинге 4.4 представлен план выполнения этого запроса.

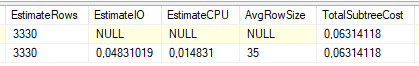
Листинг 4.3 – Запрос с низким уровнем селективности

|  |
| --- |
| SET STATISTICS PROFILE ON;  GO  USE Movie;  SELECT \* FROM Статистика S  WHERE S.Фильм\_idФильм = 12  GO  SET STATISTICS PROFILE OFF; |

Листинг 4.4 – План выполнения запроса с низким уровнем селективности

|  |
| --- |
| |--Clustered Index Scan(OBJECT:([Movie].[dbo].[Статистика].[PK\_Статистика] AS [S]), WHERE:([Movie].[dbo].[Статистика].[Фильм\_idФильм] as [S].[Фильм\_idФильм]=(12))) |

На рисунке 4.2 представлено время, затраченное на выполнения данного запроса.



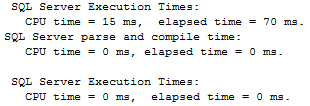


Рисунок 4.2 – Время выполнения запроса с

низким уровнем селективности

Использование кластеризованного индекса представлено в листинге 4.5. В листинге 4.6 представлен план выполнения этого запроса.

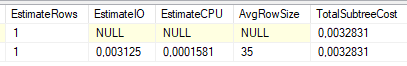
Листинг 4.5 – Запрос с использование кластеризованного индекса

|  |
| --- |
| SET SHOWPLAN\_TEXT ON;  GO  USE Movie;  SELECT \* FROM Статистика S  WHERE S.idСтатистика = 1;  GO  SET SHOWPLAN\_TEXT OFF; |

Листинг 4.6 – План выполнения запроса с кластеризованным индексом

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

На рисунке 4.3 представлено время, затраченное на выполнения данного запроса. Как видно из рисунка, в столбцах EstimateIO и EstimateCPU скорость выполнения запросов уменьшилась из-за использования кластеризованного индекса.



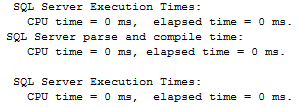


Рисунок 4.3 – Время выполнения запроса с использованием

кластеризованного индекса

В листинге 4.7 представлен запрос с условием на поле, которое не является ключом, ни первичным, ни внешним. Данный запрос захватывает 4843 записей в таблице.

Листинг 4.7 – Запрос с условием на поле

|  |
| --- |
| SET STATISTICS PROFILE ON;  GO  USE Movie;  SELECT [Кол-во\_проданных\_билетов] FROM Статистика S  WHERE S.[Кол-во\_проданных\_билетов] > 200;  GO  SET STATISTICS PROFILE OFF; |

В листинге 4.8 представлен запрос с группировкой по столбцу, которое не является ключом, ни первичным, ни внешним. Данный запрос захватывает 4849 записей в таблице.

Листинг 4.8 – Запрос с группировкой по столбцу

|  |
| --- |
| SET STATISTICS PROFILE ON;  GO  USE Movie;  SELECT Заработок FROM Статистика S  WHERE S.Заработок > 60000  GROUP BY S.Заработок  GO  SET STATISTICS PROFILE OFF; |

На рисунках 4.4 и 4.5 представлено создание статистики по столбцам, которые использовались в запросах, представленных в листингах 4.7 и 4.8.

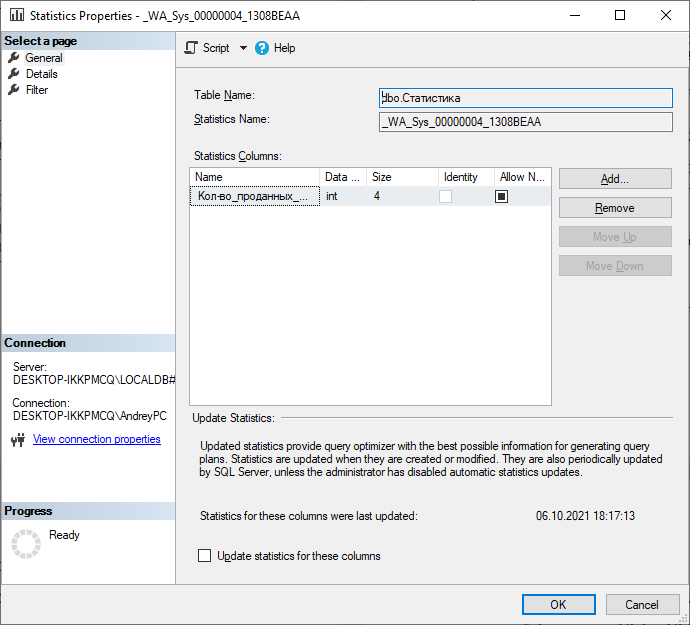


Рисунок 4.4 – Создание статистики для первого запроса

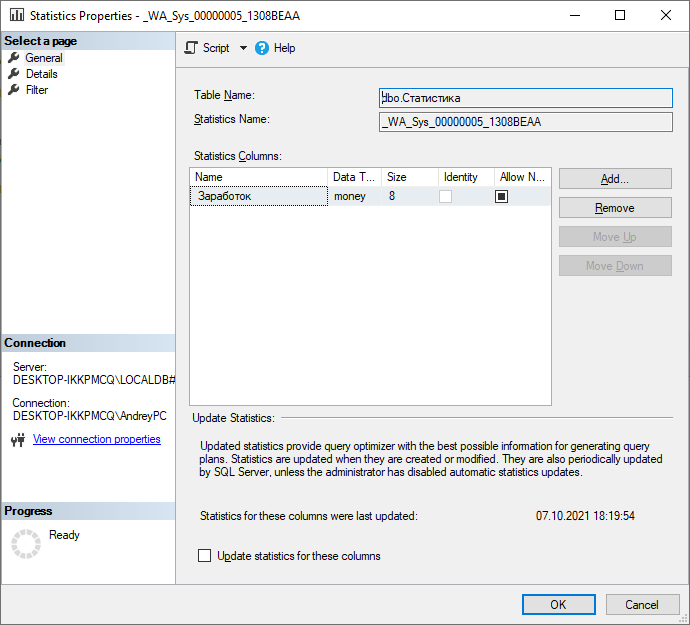
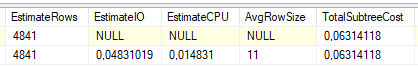


Рисунок 4.5 – Создание статистики для второго запроса

В листинге 4.9 представлен план выполнения первого запроса. На рисунке 4.6 представлено время, затраченное на выполнение первого запроса.

Листинг 4.9 – План выполнения первого запроса с статистикой

|  |
| --- |
| |--Clustered Index Scan(  OBJECT:([Movie].[dbo].[Статистика].[PK\_Статистика] AS [S]), WHERE:([Movie].[dbo].[Статистика].[Кол-во\_проданных\_билетов] as [S].[Кол-во\_проданных\_билетов]>(200))) |



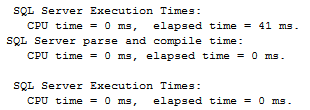
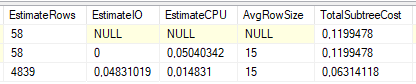


Рисунок 4.6 – Время выполнения первого запроса

В листинге 4.10 представлен план выполнения второго запроса. На рисунке 4.7 представлено время, затраченное на выполнение второго запроса.

Листинг 4.10 – План выполнения второго запроса с статистикой

|  |
| --- |
| |--Hash Match(Aggregate, HASH:([S].[Заработок]),  RESIDUAL:([Movie].[dbo].[Статистика].[Заработок] as  [S].[Заработок] = [Movie].[dbo].[Статистика].[Заработок] as [S].[Заработок]))  |--Clustered Index Scan(  OBJECT:([Movie].[dbo].[Статистика].[PK\_Статистика] AS [S]), WHERE:([Movie].[dbo].[Статистика].[Заработок] as [S].[Заработок]>($60000.0000))) |



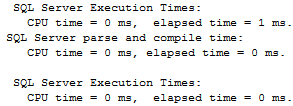


Рисунок 4.7 – Время выполнения второго запроса

В листинге 4.11 представлен код, который использовался для очищения статистики.

Листинг 4.11 – Очищение статистики

|  |
| --- |
| DROP STATISTICS Статистика.WA\_Sys\_00000002\_1308BEAA  DROP STATISTICS Статистика.WA\_Sys\_00000003\_1308BEAA  DROP STATISTICS Статистика.WA\_Sys\_00000004\_1308BEAA  DROP STATISTICS Статистика.WA\_Sys\_00000005\_1308BEAA |

На рисунке 4.8 и 4.9 представлено создание индекса по столбцам, которые использовались в запросах, представленных в листингах 4.7 и 4.8.

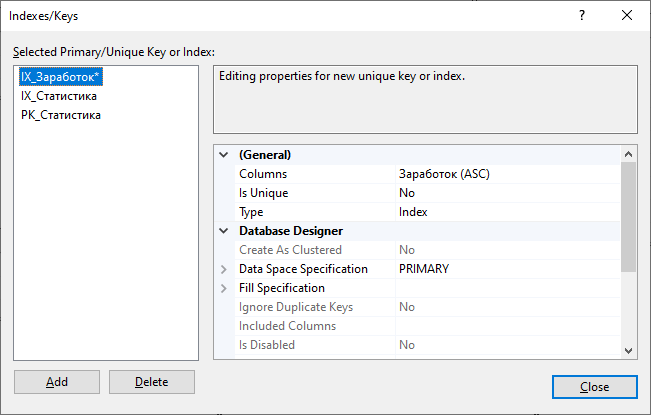


Рисунок 4.8 – Создание индекса для первого запроса

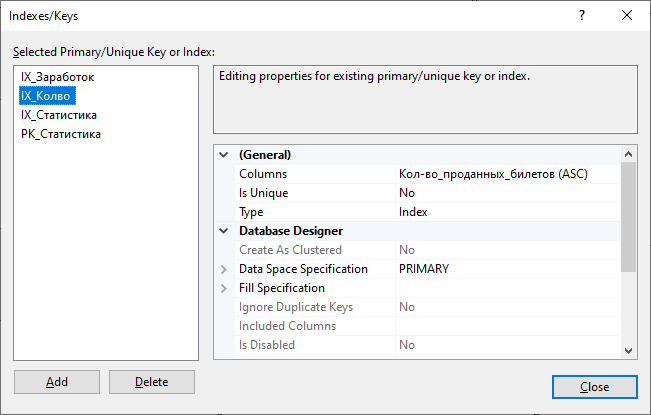
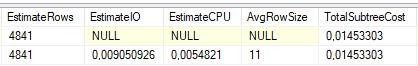


Рисунок 4.9 – Создание индекса для второго запроса

В листинге 4.12 представлен план выполнения первого запроса с использованием индекса. На рисунке 4.10 представлено время, затраченное на выполнение этого запроса.

Листинг 4.12 – План выполнения первого запроса с индексом

|  |
| --- |
| |--Index Seek(OBJECT:([Movie].[dbo].[Статистика].[IX\_Колво] AS [S]), SEEK:([S].[Кол-во\_проданных\_билетов] > CONVERT\_IMPLICIT(int,[@1],0)) ORDERED FORWARD) |



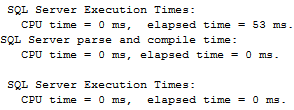
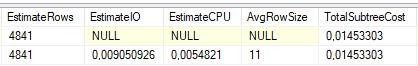


Рисунок 4.10 – Время выполнения первого запроса с индексом

В листинге 4.13 представлен план выполнения второго запроса с использованием индекса. На рисунке 4.11 представлено время, затраченное на выполнение этого запроса.

Листинг 4.13 – План выполнения второго запроса с индексом

|  |
| --- |
| |--Stream Aggregate(GROUP BY:([S].[Заработок]))  |--Index Seek(  OBJECT:([Movie].[dbo].[Статистика].[IX\_Заработок] AS [S]), SEEK:([S].[Заработок] > ($60000.0000)) ORDERED FORWARD) |



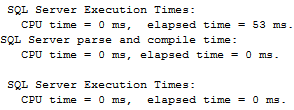


Рисунок 4.11 – Время выполнения второго запроса с индексом

**Вывод**. Если сравнивать скорость выполнения запросов при существовании индекса и без индекса, то запрос с индексом выполняется быстрее. В нашей таблицы это довольно небольшая прибавка к скорости, но это лишь из-за небольшого объема строчек в таблицы. Если таблица в базе данных будет больше, то и разность во времени выполнения запросов с индексом и без будет увеличиваться.

# 5 Выбор порядка соединения

Метод вложенного цикла основан на применении "грубой силы", или полного перебора. Иными словами, для каждой строки внешней таблицы извлекается и сравнивается каждая строка внутренней таблицы.

Соединение таблиц методом вложенного цикла представлен в листинге 5.1. План выполнения данного запроса представлен в листинге 5.2.

Листинг 5.1 – Запрос с использованием метода вложенного цикла

|  |
| --- |
| SET SHOWPLAN\_TEXT ON;  GO  USE Movie;  SELECT \* FROM Фильм F  JOIN Статистика S  ON F.idФильм = S.Фильм\_idФильм  AND F.idФильм = 6;  GO  SET SHOWPLAN\_TEXT OFF; |

Листинг 5.2 – План выполнения запроса с использованием вложенного цикла

|  |
| --- |
| |--Nested Loops(Inner Join)  |--Clustered Index Seek(  OBJECT:([Movie].[dbo].[Фильм].[PK\_\_Фильм\_\_57ED26BFD2C057FD]  AS [F]), SEEK:([F].[idФильм]=(6)) ORDERED FORWARD)  |--Clustered Index Scan(  OBJECT:([Movie].[dbo].[Статистика].[PK\_Статистика]  AS [S]), WHERE:([Movie].[dbo].[Статистика].[Фильм\_idФильм]  as [S].[Фильм\_idФильм]=(6))) |

Метод соединения хешированием хорошо подходит для незапланированных запросов, для которых не предполагается наличие индексов.

Запрос с использованием метода соединения хешированием представлен в листинге 5.3. План выполнения данного запроса представлен в листинге 5.4.

Листинг 5.3 – Запрос с использованием метода соединения хешированием

|  |
| --- |
| SET SHOWPLAN\_TEXT ON;  GO  USE Movie;  SELECT \* FROM Фильм F  JOIN Статистика S  ON F.idФильм = S.Фильм\_idФильм  GO  SET SHOWPLAN\_TEXT OFF; |

Листинг 5.4 – План выполнения запроса с использованием хеширования

|  |
| --- |
| |--Hash Match(Inner Join, HASH:([F].[idФильм])=([S].[Фильм\_idФильм]))  |--Clustered Index Scan(  OBJECT:([Movie].[dbo].[Фильм].[PK\_\_Фильм\_\_57ED26BFD2C057FD]  AS [F]))  |--Clustered Index Scan(  OBJECT:([Movie].[dbo].[Статистика].[PK\_Статистика]  AS [S])) |

# 6 Динамические представления

Применение представления sys.dm\_exec\_query\_optimizer\_info представлено в листинге 6.1. Результат выполнения данного представления представлен на рисунке 6.1.

Данное представление возвращает подробную статистику о работе оптимизатора. В столбце counter содержится имя события оптимизатора, а в столбце occurrence – отображается общее количество проявлений этих событий. Значение столбца value содержит дополнительную информацию о событиях. В столбце counter отображаются этапы процесса оптимизации.

Таким образом, в запросе исследуются, сколько раз выполняется оптимизация этапа 1.

Листинг 6.1 – Запрос для представления sys.dm\_exec\_query\_optimizer\_info

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

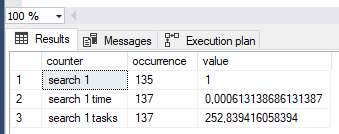


Рисунок 6.1 – Результат применения представления

sys.dm\_exec\_query\_optimizer\_info

Применение представления sys.dm\_exec\_query\_stats представлено в листинге 6.2. Результат выполнения данного представления представлен на рисунке 6.2.

Данное представление 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.

Выполнение запроса соединяет представления sys.dm\_exec\_query\_stats и sys.dm\_exec\_cached\_plans и возвращает все кэшированные планы выполнения, упорядоченные по количеству их выполнений.

Листинг 6.2 – Запрос для представления sys.dm\_exec\_query\_stats

|  |
| --- |
| 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; |

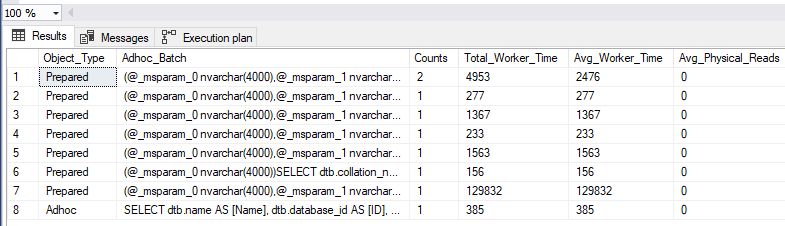


Рисунок 6.2 – Результат применения представления

sys.dm\_exec\_query\_stats

# 7 Подсказки оптимизации

В большинстве случаев оптимизатор запросов выбирает самый быстрый план выполнения. Но в некоторых особых ситуациях оптимизатор, в силу определенных обстоятельств, не может найти оптимального решения. В таких случаях следует использовать подсказки оптимизации, чтобы вынудить оптимизатор использовать определенный план выполнения с лучшей производительностью.

Принуждение оптимизатора к использованию индекса представлено в листинге 7.1. На рисунке 7.1 представлен план выполнения данного запроса.

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

Листинг 7.1 – Запрос с использованием подсказки INDEX

|  |
| --- |
| SET SHOWPLAN\_TEXT ON;  GO  USE Movie;  SELECT \* FROM Статистика S  WITH (INDEX(PK\_Статистика))  WHERE S.idСтатистика = 310;  GO  SET SHOWPLAN\_TEXT OFF; |

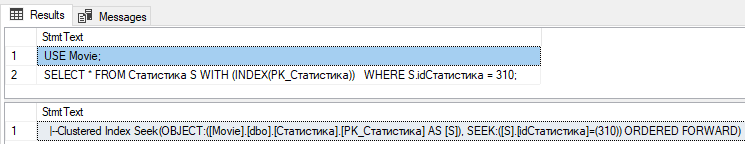


Рисунок 7.1 – План выполнения запроса с использованием

подсказки INDEX

Принуждение оптимизатора не использовать некластеризованные индексы представлено в листинге 7.2. На рисунке 7.2 представлен план выполнения данного запроса.

Запрос содержит подсказку INDEX(0), которая заставляет оптимизатор не использовать никакие существующие некластеризованные индексы.

Листинг 7.2 – Запрос с использованием подсказки INDEX(0)

|  |
| --- |
| SET SHOWPLAN\_TEXT ON;  GO  USE Movie;  SELECT \* FROM Статистика S  WITH(INDEX(0))  WHERE S.idСтатистика = 310;  GO  SET SHOWPLAN\_TEXT OFF; |

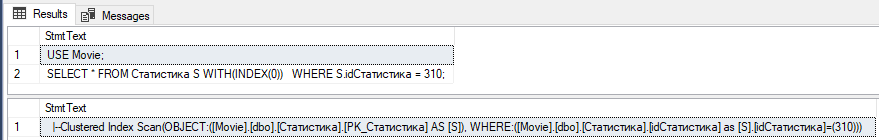


Рисунок 7.2 – План выполнения запроса с использованием

подсказки INDEX(0)

Использование подсказки FORCE ORDER представлено в листинге 7.3. На рисунке 7.3 представлен план выполнения данного запроса.

Запрос содержит подсказку FORCE ORDER, которая заставляет оптимизатор соединять таблицы в том порядке, в котором они указаны в запросе.

Листинг 7.3 – Запрос с использованием подсказки FORCE ORDER

|  |
| --- |
| SET SHOWPLAN\_TEXT ON;  GO  USE Movie;  SELECT \* FROM Фильм F JOIN Статистика S  ON F.idФильм = S.Фильм\_idФильм OPTION(FORCE ORDER);  GO  SET SHOWPLAN\_TEXT OFF; |

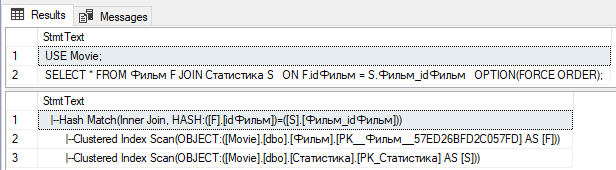


Рисунок 7.3 – План выполнения запроса с использованием

подсказки FORCE ORDER

Использование подсказки слияния MERGE представлено в листинге 7.4. На рисунке 7.4 представлен план выполнения данного запроса.

Запрос содержит подсказку слияния MERGE, которая заставляет оптимизатор использовать технику соединения слиянием.

Листинг 7.4 – Запрос с использованием подсказки MERGE

|  |
| --- |
| SET SHOWPLAN\_TEXT ON;  GO  USE Movie;  SELECT \* FROM Фильм F JOIN Статистика S  ON F.idФильм = S.Фильм\_idФильм OPTION (MERGE JOIN);  GO  SET SHOWPLAN\_TEXT OFF; |

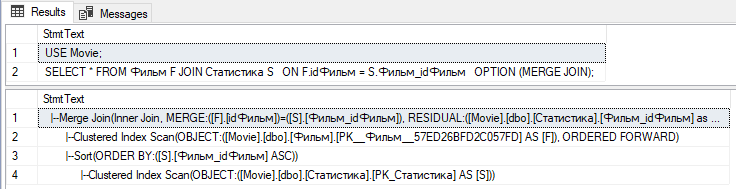


Рисунок 7.4 – План выполнения запроса с использованием

подсказки MERGE

Использование подсказки слияния MERGE в предложении FROM представлено в листинге 7.5. Запрос идентичен запросу, который был представлен в листинге 7.4, но в нем подсказка соединения указывается в предложении FROM.

Листинг 7.5 – Запрос с использованием подсказки MERGE в предложении FROM

|  |
| --- |
| SET SHOWPLAN\_TEXT ON;  GO  USE Movie;  SELECT \* FROM Фильм F  JOIN Статистика S  ON F.idФильм = S.Фильм\_idФильм  OPTION (MERGE JOIN);  GO  SET SHOWPLAN\_TEXT OFF; |

Использование подсказки запроса OPTIMIZE FOR представлено в листинге 7.6. На рисунке 7.6 представлен результат выполнения данного запроса.

Запрос содержит подсказку OPTIMIZE FOR, которая заставляет оптимизатор запросов использовать определенное значение для локальной переменной при компилировании и оптимизации запроса.

Листинг 7.6 – Запрос с использованием подсказки OPTIMIZE FOR

|  |
| --- |
| DECLARE @cost int  SET @cost = 222  SELECT \* FROM Статистика  WHERE [Кол-во\_проданных\_билетов] = @cost  OPTION (OPTIMIZE FOR (@cost = 111)); |

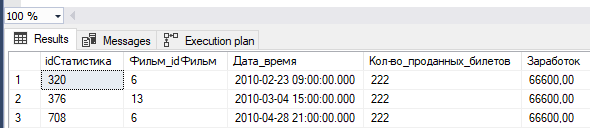


Рисунок 7.6 – План выполнения запроса с использованием

подсказки OPTIMIZE FOR

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

Создание структуры плана представлено в листинге 7.7. Данная структура аналогична запросу, представленному в листинге 7.3, соответственно, план выполнения запроса также аналогичен.

Системная процедура sp\_create\_plan\_guide принимает несколько параметров. В параметре @name указывается имя создаваемой структуры плана. Параметр @stmt содержит инструкция Transact-SQL, а параметр @type указывает тип структуры плана (SQL, OBJECT или TEMPLATE). Подсказка по оптимизации указывается в параметре @hints.

Листинг 7.7 – Запрос для создания структуры плана

|  |
| --- |
| sp\_create\_plan\_guide @name = N'Example\_19\_15',  @stmt = N'SELECT \* FROM Фильм F JOIN Статистика S ON F.idФильм = S.Фильм\_idФильм',  @type = N'SQL',  @module\_or\_batch = NULL,  @params = NULL,  @hints = N'OPTION (HASH JOIN)' |

На рисунке 7.7 представлена структура плана «Example\_19\_15», полученная с помощью представление каталога sys.plan\_guides. Это представление содержит строку для каждой структуры плана в текущей базе данных. Наиболее важными столбцами этого представления являются столбцы plan\_guid\_id, name и query\_text. Столбец plan\_guide\_id содержит однозначный идентификатор структуры плана, а столбец name определяет ее имя. В столбце query\_text указывается текст запроса, для которого предназначается данная структура плана.

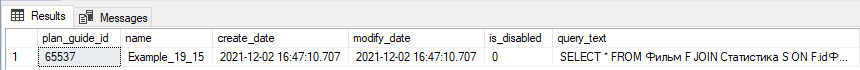


Рисунок 7.7 – Структура плана «Example\_19\_15»

# 8 Настройка производительности

Использование представления каталога sys.sysprocesses представлено в листинге 8.1. Результат выполнения данного запроса представлен на рисунке 8.1.

Данное представление содержит информацию о процессах, выполняющихся на экземпляре сервера. spid (идентификатор сессии), dbid (идентификатор текущей базы данных), uid (идентификатор пользователя, выполняющего текущую команду) и cpu (общее время центрального процессора, затраченного на данный процесс).

Листинг 8.1 – Запрос с использованием представления каталога sys.sysprocesses

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

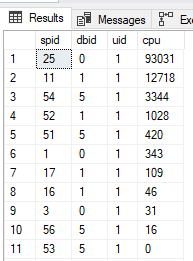


Рисунок 8.1 – Результат выполнения запроса с использованием

представления каталога sys.sysprocesses

Использование динамического административного представления sys.dm\_exec\_requests представлено в листинге 8.2. Результат выполнения данного запроса представлен на рисунке 8.2.

Данное представление предоставляет ту же информацию, что и представление каталога sys.sysprocesses, но его столбцы называются по-другому. Столбец sql\_handle этого представления указывает на область, в которой хранится весь пакет.

Листинг 8.2 – Запрос с использованием представления sys.dm\_exec\_requests

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

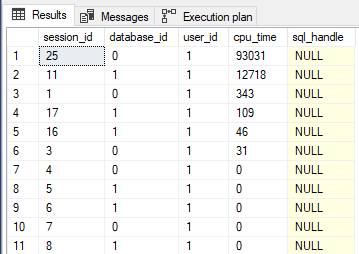


Рисунок 8.2 – Результат выполнения запроса с использованием

представления sys.dm\_exec\_requests

Использование динамического административного представления sys.dm\_exec\_query\_stats представлено в листинге 8.3. Результат выполнения данного запроса представлен на рисунке 8.3.

Данное представление отображает информацию о кэшированных инструкциях Transact-SQL и хранимых процедурах, которые используют больше всего времени центрального процессора. В столбце total\_worker\_time представления sys.dm\_exec\_query\_stats 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.

Листинг 8.3 – Запрос с использованием представления sys.dm\_exec\_query\_stats

|  |
| --- |
| 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; |

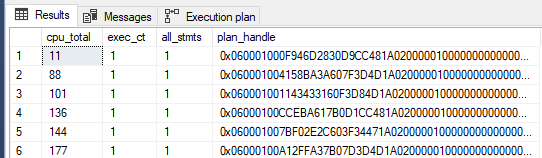


Рисунок 8.3 – Результат выполнения запроса с использованием

представления sys.dm\_exec\_query\_stats

Использование представления sys.dm\_os\_memory\_clerks представлено в листинге 8.4. Результат выполнения данного запроса представлен на рисунке 8.4.

Данное представление 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.

Листинг 8.4 – Запрос с использованием представления sys.dm\_os\_memory\_clerks

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

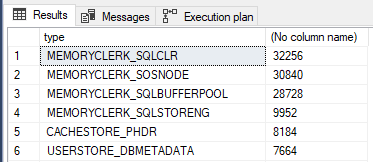


Рисунок 8.4 – Результат выполнения запроса с использованием

представления sys.dm\_os\_memory\_clerks

Использование динамического административного представления sys.dm\_os\_memory\_objects представлено в листинге 8.5. Результат выполнения данного запроса представлен на рисунке 8.5.

Данное представление возвращает объекты памяти, которые в настоящее время распределены системой баз данных. В основном используется для анализа использования памяти и для определения возможного недостатка памяти. Запрос группирует все объекты памяти по их типу, а затем использует значения в столбце pages\_in\_bytes для отображения общего объема памяти в каждой группе.

Листинг 8.5 – Запрос с использованием sys.dm\_os\_memory\_objects

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

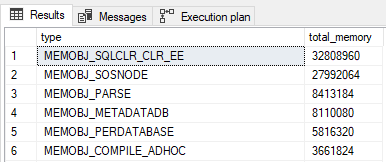


Рисунок 8.5 – Результат выполнения запроса с использованием

представления sys.dm\_os\_memory\_objects

Использование динамического административного представления sys.dm\_os\_wait\_stats представлено в листинге 8.6. Результат выполнения данного запроса представлен на рисунке 8.6.

Данное представление возвращает информацию об ожиданиях в выполняемых потоках. можно использовать для диагностики проблем производительности как всего сервера Database Engine, так и отдельных запросов и пакетов. Первый столбец содержит названия типов ожидания, а второй – количество ожиданий соответствующего типа.

Листинг 8.6 – Запрос с использованием представления sys.dm\_os\_wait\_stats

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

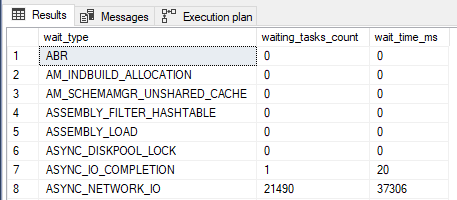


Рисунок 8.6 – Результат выполнения запроса с использованием

представления sys.dm\_os\_wait\_stats

Использование динамического административного представления sys.dm\_io\_virtual\_file\_stats представлено в листинге 8.7. Результат выполнения данного запроса представлен на рисунке 8.7.

Данное представление возвращает статистику ввода/вывода для файлов данных и журнала. В первом параметре, database\_id, указывается однозначный идентификационный номер базы данных, а во втором, file\_id, – идентификатор файла. Когда указано значение NULL, тогда возвращаются все базы данных, т. е. все файлы экземпляра Database Engine.

Листинг 8.7 – Запрос с использованием sys.dm\_io\_virtual\_file\_stats

|  |
| --- |
| 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); |

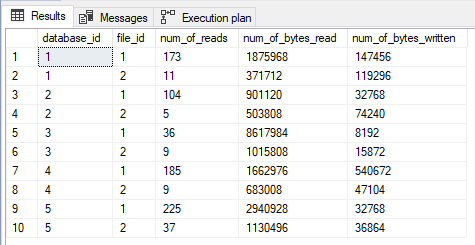


Рисунок 8.7 – Результат выполнения запроса с использованием

представления sys.dm\_io\_virtual\_file\_stats

Отображение сетевого протокола и механизма проверки подлинности представлено в листинге 8.8. Результат выполнения данного запроса представлен на рисунке 8.8.

Данный запрос возвращает основную информацию о текущем соединении: сетевой транспортный протокол и механизм аутентификации.

Листинг 8.8 – Запрос для получения информации о текущем соединении

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

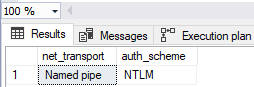


Рисунок 8.8 – Результат выполнения запроса для получения

информации о текущем соединении

Отображение количества пакетов чтения и записи num\_reads и num\_writes представлено в листинге 8.9. Результат выполнения данного запроса представлен на рисунке 8.9.

В первом столбце отображается количество прочитанных пакетов в текущем соединении, а во втором – количество записанных пакетов.

Листинг 8.9 – Запрос для отображения количества пакетов чтения и записи

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

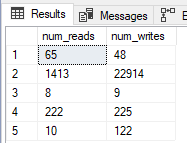


Рисунок 8.9 – Результат выполнения запроса для отображения

количества пакетов чтения и записи

Создание двух новых таблиц для базы данных sample представлено в листинге 8.10. Результат выполнения данного запроса представлен на рисунке 8.10.

Листинг 8.10 – Запрос для создания новых таблиц для базы данных sample

|  |
| --- |
| 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); |



Рисунок 8.10 – Результат выполнения запроса для создания

новых таблиц для базы данных sample

Вставка новых строк в таблицу orders представлено в листинге 8.11. Результат выполнения данного запроса представлен на рисунке 8.11.

Листинг 8.11 – Запрос для вставки новых строк в таблицу orders

|  |
| --- |
| 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 |

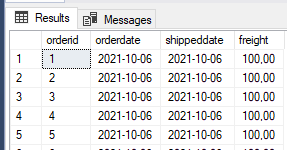


Рисунок 8.11 – Результат выполнения запроса для вставки

новых строк в таблицу orders

Вставка новых строк в таблицу order\_details представлено в листинге 8.12. Результат выполнения данного запроса представлен на рисунке 8.12.

Листинг 8.12 – Запрос для вставки новых строк в таблицу order\_details

|  |
| --- |
| 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) |

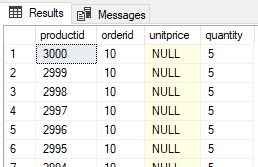


Рисунок 8.12 – Результат выполнения запроса для вставки

новых строк в таблицу order\_details

Запрос для трассировки представлен в листинге 8.13. Результат выполнения данного запроса представлен на рисунке 8.13.

Листинг 8.13 – Запрос для трассировки

|  |
| --- |
| 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); |

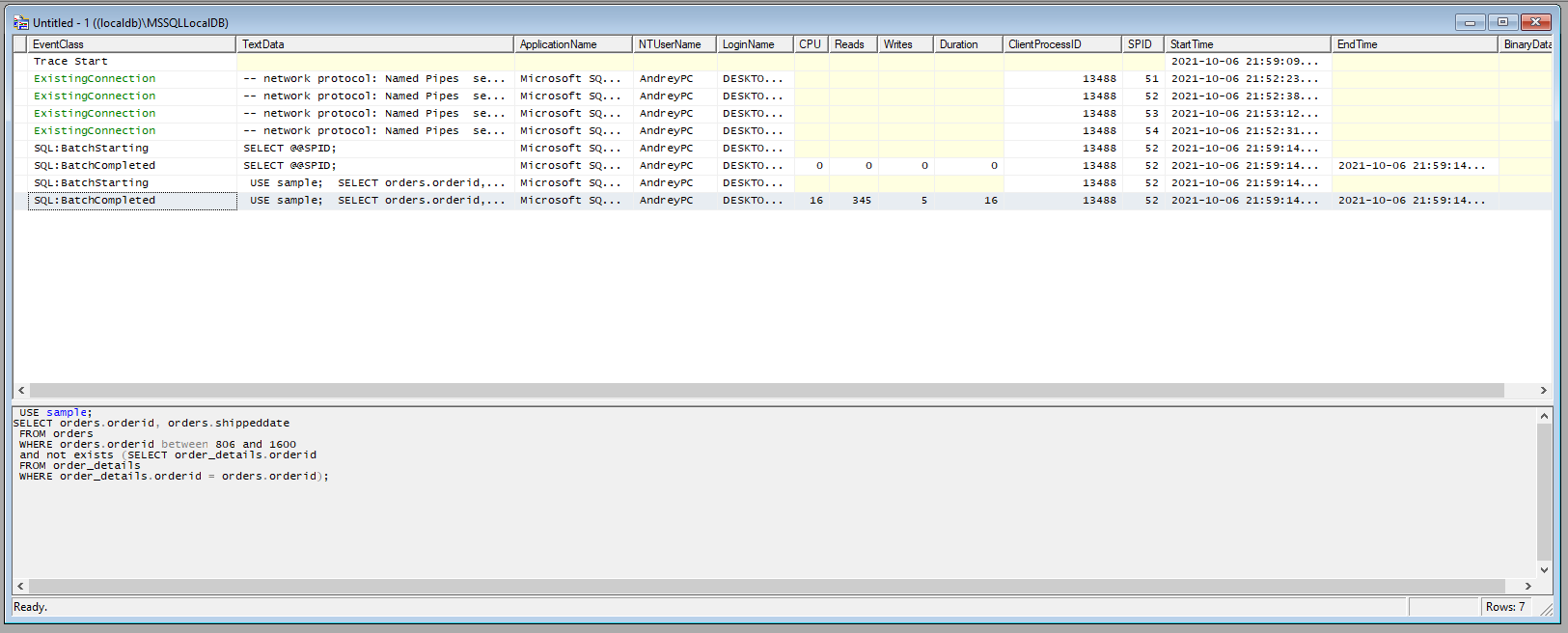


Рисунок 8.13 – Результат выполнения запроса для трассировки

# Заключение

Оптимизатор запросов является составной частью компонента Database Engine, которая определяет наиболее оптимальный метод выполнения запроса. Оптимизатор создает несколько планов выполнения для данного запроса и выбирает из них наименее затратный.

Процесс оптимизации запроса можно разбить на следующие этапы: анализ запроса, выбор индексов и выбор порядка выполнения соединения. В процессе анализа запроса оптимизатор исследует его на наличие аргументов поиска, использование оператора OR и наличие критериев соединения, в приведенном порядке. Идентификация аргументов поиска позволяет оптимизатору решить, использовать ли существующие индексы.

Порядок указания в предложении FROM инструкции SELECT двух или больше соединяемых таблиц не влияет на принимаемое оптимизатором решение относительно порядка их обработки. Компонент Database Engine поддерживает три различные техники обработки соединения, которые могут быть использованы оптимизатором. То, какую технику выберет оптимизатор, зависит от имеющихся статистических данных для соединяемых таблиц.

Оказать воздействие на работу оптимизатора можно, используя подсказки оптимизации. Компонент Database Engine поддерживает разнообразные подсказки оптимизации, которые можно сгруппировать следующим образом: табличные подсказки, подсказки соединения и подсказки запросов.

В данной работе была изучена и реализована тема анализа, плана выполнения запроса, также было проанализировано влияние индексов и статистики на скорость выполнения запросов. Были разобраны порядки соединения – метод вложенного цикла и соединения хэшированием, динамические представления, подсказки оптимизации и настройка производительности.

# Список использованных источников

1 Петкович, Д. Microsoft SQL Server 2012. Руководство для начинающих / Д. Петкович; Пер. с англ. – СПб.: БХВ-Петербург, 2013. – 816 с.